[**第208集 11**](#_Toc438808403)

[**注解（Annotation）： 11**](#_Toc438808404)

[**内置注解： 11**](#_Toc438808405)

[**第210集 13**](#_Toc438808406)

[**自定义注解，ORM，反射的应用： 13**](#_Toc438808407)

[**自定义注解 13**](#_Toc438808408)

[**ORM（对象关系映射）： 14**](#_Toc438808409)

[**第211集 15**](#_Toc438808410)

[**反射的初步： 16**](#_Toc438808411)

[**Bean代码： 16**](#_Toc438808412)

[**测试代码： 17**](#_Toc438808413)

[**第212集 19**](#_Toc438808414)

[**利用反射的API，来获取类的信息： 19**](#_Toc438808415)

[**通过类的模板获取类的，属性，方法，构造器，（有参，无参）！ 19**](#_Toc438808416)

[**反射测试2： 21**](#_Toc438808417)

[**第213集 23**](#_Toc438808418)

[**反射操作注解，反射操作泛型： 24**](#_Toc438808419)

[**操作泛型的代码如下： 24**](#_Toc438808420)

[**反射效率比较，以及设置SetAccessible(true)后的效率对比代码： 27**](#_Toc438808421)

[**操作注解（Annotation）的代码如下： 29**](#_Toc438808422)

[**第214集 32**](#_Toc438808423)

[**动态编译： 32**](#_Toc438808424)

[**实例代码如下： 32**](#_Toc438808425)

[**第215集 34**](#_Toc438808426)

[**动态调用javaScript代码： 34**](#_Toc438808427)

[**js代码如下： 36**](#_Toc438808428)

[**第216集 37**](#_Toc438808429)

[**java语言的动态性: 37**](#_Toc438808430)

[**javassist入门： 37**](#_Toc438808431)

[**emp类： 39**](#_Toc438808432)

[**第217集 40**](#_Toc438808433)

[**Javassist进阶： 40**](#_Toc438808434)

[**API的用法详解： 40**](#_Toc438808435)

[**第220集 46**](#_Toc438808436)

[**深入类类加载 47**](#_Toc438808437)

[**双亲委托机制: 47**](#_Toc438808438)

[**第221集 48**](#_Toc438808439)

[**自定义类加载器深入： 48**](#_Toc438808440)

[**文件类加载器： 48**](#_Toc438808441)

[**测试类的代码如下： 51**](#_Toc438808442)

[**网络类加载器： 53**](#_Toc438808443)

[**第222集 56**](#_Toc438808444)

[**类加载器之加密技术初步： 56**](#_Toc438808445)

[**自定义加密解密类加载器技术展示的代码如下： 56**](#_Toc438808446)

[**加密类加载器代码： 57**](#_Toc438808447)

[**解密的类加载器代码如下： 59**](#_Toc438808448)

[**第223集 62**](#_Toc438808449)

[**线程上下文类加载器： 63**](#_Toc438808450)

[**第224集 64**](#_Toc438808451)

[**内部类技术 64**](#_Toc438808452)

[**内部类的存在意义： 64**](#_Toc438808453)

[**第225集 68**](#_Toc438808454)

[**方法内部类： 69**](#_Toc438808455)

[**匿名内部类: 70**](#_Toc438808456)

[**第226集 72**](#_Toc438808457)

[**GOF23设计模式（group of four ） 72**](#_Toc438808458)

[**第227集 74**](#_Toc438808459)

[**单例模式之双重检测锁实现： 75**](#_Toc438808460)

[**静态内部类实现单例模式： 76**](#_Toc438808461)

[**枚举方式实现单例模式： 77**](#_Toc438808462)

[**第228集 78**](#_Toc438808463)

[**单例模式如何防止反射和反序列化的漏洞！ 78**](#_Toc438808464)

[**测试反射和反序列化的漏洞 79**](#_Toc438808465)

[**测试单例的5种情况的效率： 81**](#_Toc438808466)

[**第229集 83**](#_Toc438808467)

[**工厂模式 83**](#_Toc438808468)

[**实现类Audi 84**](#_Toc438808469)

[**实现类Byd 84**](#_Toc438808470)

[**测试类Client01,没有工厂模式的时候： 85**](#_Toc438808471)

[**添加一个工厂类CarFactory 85**](#_Toc438808472)

[**工厂类也可以这样 86**](#_Toc438808473)

[**测试类 87**](#_Toc438808474)

[**第230集 88**](#_Toc438808475)

[**工厂模式 88**](#_Toc438808476)

[**汽车接口： 88**](#_Toc438808477)

[**汽车工厂接口： 89**](#_Toc438808478)

[**车的实现类如下： 89**](#_Toc438808479)

[**测试类 91**](#_Toc438808480)

[**增加一个实现类 91**](#_Toc438808481)

[**简单工厂，工厂方法： 92**](#_Toc438808482)

[**第231集 93**](#_Toc438808483)

[**抽象工厂模式： 93**](#_Toc438808484)

[**接口 94**](#_Toc438808485)

[**汽车工厂接口： 96**](#_Toc438808486)

[**测试代码： 98**](#_Toc438808487)

[**第232集 98**](#_Toc438808488)

[**建造者模式： 99**](#_Toc438808489)

[**第233-234集 104**](#_Toc438808490)

[**原型模式 104**](#_Toc438808491)

[**浅克隆： 104**](#_Toc438808492)

[**深克隆 107**](#_Toc438808493)

[**测试： 108**](#_Toc438808494)

[**深克隆的另一种实现方式，序列化和反序列化实现身： 109**](#_Toc438808495)

[**测试类 111**](#_Toc438808496)

[**第235集 112**](#_Toc438808497)

[**适配器模式 113**](#_Toc438808498)

[**第238集 116**](#_Toc438808499)

[**桥接模式： 116**](#_Toc438808500)

[**商品类 117**](#_Toc438808501)

[**测试： 120**](#_Toc438808502)

[**使用桥接模式后的测试代码如下： 122**](#_Toc438808503)

[**第239集 122**](#_Toc438808504)

[**组合模式（composite）: 123**](#_Toc438808505)

[**应用场景： 123**](#_Toc438808506)

[**组合模式核心： 123**](#_Toc438808507)

[**使用组合模式，模拟杀毒软件的架构设计！ 124**](#_Toc438808508)

[**测试 127**](#_Toc438808509)

[**第240集 127**](#_Toc438808510)

[**装饰模式（decorator）： 128**](#_Toc438808511)

[**代码的展示如下： 129**](#_Toc438808512)

[**测试代码如下： 132**](#_Toc438808513)

[**第241集 133**](#_Toc438808514)

[**门面模式（外观模式）： 133**](#_Toc438808515)

[**代码展示如下： 133**](#_Toc438808516)

[**门面模式的展示代码如下： 135**](#_Toc438808517)

[**测试代码： 136**](#_Toc438808518)

[**第242集 137**](#_Toc438808519)

[**享元模式（FlyWeight）： 137**](#_Toc438808520)

[**核心： 137**](#_Toc438808521)

[**代码展示： 137**](#_Toc438808522)

[**测试代码： 140**](#_Toc438808523)

[**第243集 141**](#_Toc438808524)

[**责任链模式（chain of responsibility）： 141**](#_Toc438808525)

[**代码展示如下： 142**](#_Toc438808526)

[**测试代码如下： 148**](#_Toc438808527)

[**第244集 148**](#_Toc438808528)

[**迭代器模式（Iterator）： 149**](#_Toc438808529)

[**迭代器接口： 151**](#_Toc438808530)

[**测试代码： 152**](#_Toc438808531)

[**第245集 153**](#_Toc438808532)

[**中介者模式（Mediator）： 153**](#_Toc438808533)

[**第246集 157**](#_Toc438808534)

[**命令模式（command）: 157**](#_Toc438808535)

[**结构： 158**](#_Toc438808536)

[**代码展示如下： 158**](#_Toc438808537)

[**第247集 160**](#_Toc438808538)

[**解释器模式（Interpreter）： 161**](#_Toc438808539)

[**开发中常见的场景: 161**](#_Toc438808540)

[**第248集 162**](#_Toc438808541)

[**策略模式（strategy）： 162**](#_Toc438808542)

[**接口的实现类如下： 164**](#_Toc438808543)

[**第249集 167**](#_Toc438808544)

[**模板方法模式（template method）: 167**](#_Toc438808545)

[**第250集 169**](#_Toc438808546)

[**状态模式（State）： 169**](#_Toc438808547)

[**接口的实现类如下： 170**](#_Toc438808548)

[**测试类： 172**](#_Toc438808549)

[**第251集 173**](#_Toc438808550)

[**观察者模式（Observer） 173**](#_Toc438808551)

[**测试类代码如下： 175**](#_Toc438808552)

[**第252集 176**](#_Toc438808553)

[**观察者模式进阶（jdk）: 177**](#_Toc438808554)

[**测试类 179**](#_Toc438808555)

[**第253集 180**](#_Toc438808556)

[**备忘录模式（memento） 180**](#_Toc438808557)

[**核心： 180**](#_Toc438808558)

[**结构： 180**](#_Toc438808559)

[**备忘点较多时： 180**](#_Toc438808560)

[**测试类： 184**](#_Toc438808561)

[**第254集 185**](#_Toc438808562)

[**正则表达式基本知识： 185**](#_Toc438808563)

[**基本语法 185**](#_Toc438808564)

[**高级语法 185**](#_Toc438808565)

[**第255集 188**](#_Toc438808566)

[**正则表达式 188**](#_Toc438808567)

[**第257集 190**](#_Toc438808568)

[**正则语法之选择符和分组以及反向引用 190**](#_Toc438808569)

[**第258集 191**](#_Toc438808570)

[**正则语法7: 191**](#_Toc438808571)

[**第259集 192**](#_Toc438808572)

[**正则的练习： 193**](#_Toc438808573)

[**常见的正则的表达式： 193**](#_Toc438808574)

[**第260集 194**](#_Toc438808575)

[**正则的使用： 194**](#_Toc438808576)

[**第261集 195**](#_Toc438808577)

[**Java程序中如何使用正则表达式： 195**](#_Toc438808578)

[**正则对象中分组的处理： 197**](#_Toc438808579)

[**正则对象的替换操作 198**](#_Toc438808580)

[**正则对象对分割字符串的操作： 199**](#_Toc438808581)

[**第262集 199**](#_Toc438808582)

[**正则对象的应用 200**](#_Toc438808583)

[**第263集 202**](#_Toc438808584)

[**JDBC数据库的操作： 202**](#_Toc438808585)

[**Mysql的下载： 204**](#_Toc438808586)

[**第264集 204**](#_Toc438808587)

[**Mysql的客户端软件Navicat软件的使用： 204**](#_Toc438808588)

[**Navicat的常见操作，请观看"三百集"视频即可。 205**](#_Toc438808589)

[**第265集 205**](#_Toc438808590)

[**mysql数据库的命令的行操作： 205**](#_Toc438808591)

[**常见的操作如下： 206**](#_Toc438808592)

[**第266集 206**](#_Toc438808593)

[**JDBC技术（Java Database Connection）； 207**](#_Toc438808594)

[**连接数据的代码如下： 207**](#_Toc438808595)

[**第267集 209**](#_Toc438808596)

[**JDBC技术 209**](#_Toc438808597)

[**第268集 211**](#_Toc438808598)

[**JDBC技术 211**](#_Toc438808599)

[**第269集 214**](#_Toc438808600)

[**JDBC技术 214**](#_Toc438808601)

[**ResultSet结果集的处理： 214**](#_Toc438808602)

[**第270集 216**](#_Toc438808603)

[**JDBC批处理技术： 216**](#_Toc438808604)

[**第271集 219**](#_Toc438808605)

[**JDBC当中事务的概念： 219**](#_Toc438808606)

[**事务的四大特点（ACID） 220**](#_Toc438808607)

[**事务隔离级别从低到高： 220**](#_Toc438808608)

[**事务的有关代码操作如下： 220**](#_Toc438808609)

[**第272集 223**](#_Toc438808610)

[**JDBC技术当中的时间类型 223**](#_Toc438808611)

[**1） java.util.Date 223**](#_Toc438808612)

[**2）日期比较处理 223**](#_Toc438808613)

[**第273集 225**](#_Toc438808614)

[**JDBC之时间类的应用 226**](#_Toc438808615)

[**查某段时间的数据的实现： 226**](#_Toc438808616)

[**字符串--》时间类的转换： 226**](#_Toc438808617)

[**第274集 229**](#_Toc438808618)

[**JDBC之Clob（Charactor Large Object）文本存储大数据对象： 229**](#_Toc438808619)

[**-CLOB用于存储大量的文本数据 229**](#_Toc438808620)

[**Mysql中相关类型如下： 229**](#_Toc438808621)

[**代码展示如下： 229**](#_Toc438808622)

[**第275集 232**](#_Toc438808623)

[**JDBC之Bloc对象的使用： 232**](#_Toc438808624)

[**关于Blob的代码演示如下： 233**](#_Toc438808625)

[**第276集 236**](#_Toc438808626)

[**JDBC的总结以封装： 236**](#_Toc438808627)

[**配置文件的代码为： 236**](#_Toc438808628)

[**JDBCUtil工具类的代码： 237**](#_Toc438808629)

[**第277集 240**](#_Toc438808630)

[**ORM（Object Relationship Mapping）的基本思想： 240**](#_Toc438808631)

[**代码的展示 241**](#_Toc438808632)

[**第278集 243**](#_Toc438808633)

[**JDBC技术的应用： 243**](#_Toc438808634)

[**封装到map中的代码演示如下： 243**](#_Toc438808635)

[**第279集 247**](#_Toc438808636)

[**JDBC技术之将一条记录封装的javaBean中 247**](#_Toc438808637)

[**Emp类的封装代码如下： 248**](#_Toc438808638)

[**javaBean Dept类的封装如下： 250**](#_Toc438808639)

[**测试代码如下： 252**](#_Toc438808640)

[**第280集 254**](#_Toc438808641)

[**SORM（simple Object relationship Mapping）的基本思想及框架的结构： 254**](#_Toc438808642)

[**SORM的架构 255**](#_Toc438808643)

[**第281-282集 257**](#_Toc438808644)

[**对SORM进行规划设计： 257**](#_Toc438808645)

[**这两集主要是封装了Configuration类和TableContext类： 258**](#_Toc438808646)

[**TableContext类： 261**](#_Toc438808647)

[**DBManager类代码如下： 265**](#_Toc438808648)

[**第283集 268**](#_Toc438808649)

[**SORM的进阶： 268**](#_Toc438808650)

[**第284集 270**](#_Toc438808651)

[**SORM进阶： 270**](#_Toc438808652)

[**JavaFieldGetSet类 271**](#_Toc438808653)

[**JavaFileUtils类代码 272**](#_Toc438808654)

[**第286集 278**](#_Toc438808655)

[**SORM进阶： 278**](#_Toc438808656)

[**第287集 283**](#_Toc438808657)

[**ＳＯＲＭ查询Query接口的实现类的继续封装： 283**](#_Toc438808658)

[**mysqlQuery类的代码如下： 283**](#_Toc438808659)

[**reflectUtils类： 292**](#_Toc438808660)

[**第289-290集 293**](#_Toc438808661)

[**ＳＯＲＭ进阶: 293**](#_Toc438808662)

[**第291-292集 302**](#_Toc438808663)

[**SORM进阶： 302**](#_Toc438808664)

[**mysqlQuery当中的querRows方法的封装： 302**](#_Toc438808665)

[**EmpVo对象的封装： 311**](#_Toc438808666)

[**第293集 313**](#_Toc438808667)

[**SORM进阶： 313**](#_Toc438808668)

[**第294-297集 321**](#_Toc438808669)

[**SORM代码的优化： 322**](#_Toc438808670)

[**Query.类的进一步优化： 322**](#_Toc438808671)

[**CallBack类封装 331**](#_Toc438808672)

[**MySqlQuery的进一步优化： 332**](#_Toc438808673)

[**第298集 336**](#_Toc438808674)

[**SORM项目的整理导出： 336**](#_Toc438808675)

[**第299集 337**](#_Toc438808677)

[**SORM的测试： 337**](#_Toc438808678)

[**测试的代码如下： 337**](#_Toc438808679)

[**po类代码 338**](#_Toc438808680)

[**测试功能的代码如下： 340**](#_Toc438808681)

[**第300集 344**](#_Toc438808682)

[**SORM的测试的增加 344**](#_Toc438808683)

# 第208集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季208集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 注解（Annotation）：

## 内置注解：

        @Override对父类的方法重写：

        @Deprecated过时的，废弃的：

        @SuppressWarning 镇压报警信息

package com.bjsxt.test.annotation;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

/\*\*

 \* 测试内置注解用法

 \* @author 尚学堂高淇

 \*

 \*/

@SuppressWarnings("all")

public class Demo01  /\*extends Object\*/ {

@Override

public String toString(){

return "";

}

@Deprecated

public static void test001(){

System.out.println("test001");

}

public static void test002(){

List list = new ArrayList();

List list2 = new ArrayList();

}

public static void main(String[] args) {

Date d = new Date();

test001();

}

}

# 第210集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季210集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 自定义注解，ORM，反射的应用：

## 自定义注解

package com.bjsxt.test.annotation;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Target(value={ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

public @interface SxtTable {

String value();

}

package com.bjsxt.test.annotation;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Target(value={ElementType.FIELD})

@Retention(RetentionPolicy.RUNTIME)

public @interface SxtField {

String columnName();

String type();

int length();

}

## ORM（对象关系映射）：

package com.bjsxt.test.annotation;

@SxtTable("tb\_student")

public class SxtStudent {

@SxtField(columnName="id",type="int",length=10)

private int id;

@SxtField(columnName="sname",type="varchar",length=10)

private String studentName;

@SxtField(columnName="age",type="int",length=3)

private int age;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getStudentName() {

return studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

# 第211集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季211集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 反射的初步：

    动态语言： 程序运行时，可以改变程序结构或变量类型，如：Python,ruby,javascript等等。

## Bean代码：

package com.bjsxt.test.bean;

public class User {

private int id;

private int age;

private String uname;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getUname() {

return uname;

}

public void setUname(String uname) {

this.uname = uname;

}

public void setUname() {

this.uname = "高淇";

}

public User(int id, int age, String uname) {

super();

this.id = id;

this.age = age;

this.uname = uname;

}

//javabean必须要有无参的构造方法！

public User() {

}

}

## 测试代码：

    通过反射获取类的方式展示如下：

package com.bjsxt.test;

/\*\*

 \* 测试各种类型(class,interface,enum,annotation,primitive type,void)对应的java.lang.Class对象的获取方式

 \* @author 尚学堂高淇    www.sxt.cn

 \*

 \*/

@SuppressWarnings("all")

public class Demo01 {

public static void main(String[] args) {

String path = "com.bjsxt.test.bean.User";

try {

Class clazz = Class.forName(path);

//对象是表示或封装一些数据。  一个类被加载后，JVM会创建一个对应该类的Class对象，类的整个结构信息会放到对应的Class对象中。

//这个Class对象就像一面镜子一样，通过这面镜子我可以看到对应类的全部信息。

System.out.println(clazz.hashCode());

Class clazz2 = Class.forName(path); //一个类只对应一个Class对象

System.out.println(clazz2.hashCode());

Class strClazz = String.class;

Class strClazz2 = path.getClass();

System.out.println(strClazz==strClazz2);

Class intClazz =int.class;

int[] arr01 = new int[10];

int[][] arr02 = new int[30][3];

int[] arr03 = new int[30];

double[] arr04 = new double[10];

System.out.println(arr01.getClass().hashCode());

System.out.println(arr02.getClass().hashCode());

System.out.println(arr03.getClass().hashCode());

System.out.println(arr04.getClass().hashCode());

} catch (Exception e) {

e.printStackTrace();

}

}

}

# 第212集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季212集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 利用反射的API，来获取类的信息：

## 通过类的模板获取类的，属性，方法，构造器，（有参，无参）！

package com.bjsxt.test;

import java.lang.reflect.Constructor;

import java.lang.reflect.Field;

import java.lang.reflect.Method;

/\*\*

 \* 应用反射的API，获取类的信息(类的名字、属性、方法、构造器等)

 \* @author 尚学堂高淇  www.sxt.cn

 \*

 \*/

public class Demo02 {

public static void main(String[] args) {

String path = "com.bjsxt.test.bean.User";

try {

Class clazz = Class.forName(path);

//获取类的名字

System.out.println(clazz.getName());//获得包名+类名：com.bjsxt.test.bean.User

System.out.println(clazz.getSimpleName()); //获的类名：User

//获取属性信息

// Field[] fields = clazz.getFields(); //只能获得public的field

Field[] fields = clazz.getDeclaredFields();//获得所有的field

Field f = clazz.getDeclaredField("uname");

System.out.println(fields.length);

for(Field temp:fields){

System.out.println("属性："+temp);

}

//获取方法信息

Method[] methods = clazz.getDeclaredMethods();

Method m01 = clazz.getDeclaredMethod("getUname", null);

//如果方法有参，则必须传递参数类型对应的class对象

Method m02 = clazz.getDeclaredMethod("setUname", String.class);

for(Method m:methods){

System.out.println("方法："+m);

}

//获得构造器信息

Constructor[] constructors = clazz.getDeclaredConstructors();

Constructor c = clazz.getDeclaredConstructor(int.class,int.class,String.class);

System.out.println("获得构造器："+c);

for(Constructor temp:constructors){

System.out.println("构造器："+temp);

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

## 反射测试2：

package com.bjsxt.test;

import java.lang.reflect.Constructor;

import java.lang.reflect.Field;

import java.lang.reflect.Method;

import com.bjsxt.test.bean.User;

/\*\*

 \* 通过反射API动态的操作：构造器、方法、属性

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo03 {

public static void main(String[] args) {

String path = "com.bjsxt.test.bean.User";

try {

Class<User> clazz = (Class<User>) Class.forName(path);

//通过反射API调用构造方法，构造对象

User u = clazz.newInstance(); //其实是调用了User的无参构造方法

System.out.println(u);

Constructor<User> c = clazz.getDeclaredConstructor(int.class,int.class,String.class);

User u2 = c.newInstance(1001,18,"高淇二");

System.out.println(u2.getUname());

//通过反射API调用普通方法

User u3 = clazz.newInstance();

Method method = clazz.getDeclaredMethod("setUname", String.class);

method.invoke(u3, "高淇三");   //u3.setUname("高淇三");

System.out.println(u3.getUname());

//通过反射API操作属性

User u4 = clazz.newInstance();

Field f = clazz.getDeclaredField("uname");

f.setAccessible(true); //这个属性不需要做安全检查了，可以直接访问

f.set(u4, "高淇四"); //通过反射直接写属性

System.out.println(u4.getUname()); //通过反射直接读属性的值

System.out.println(f.get(u4));

} catch (Exception e) {

e.printStackTrace();

}

}

}

# 第213集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季213集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

[ 专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 反射操作注解，反射操作泛型：

## 操作泛型的代码如下：

package com.bjsxt.test;

import java.lang.reflect.Method;

import java.lang.reflect.ParameterizedType;

import java.lang.reflect.Type;

import java.util.List;

import java.util.Map;

import com.bjsxt.test.bean.User;

/\*\*

 \* 通过反射获取泛型信息

 \* @author dell

 \*

 \*/

public class Demo04 {

public void test01(Map<String,User> map,List<User> list){

System.out.println("Demo04.test01()");

}

public Map<Integer,User> test02(){

System.out.println("Demo04.test02()");

return null;

}

public static void main(String[] args) {

try {

//获得指定方法参数泛型信息

Method m = Demo04.class.getMethod("test01", Map.class,List.class);

Type[] t = m.getGenericParameterTypes();

for (Type paramType : t) {

System.out.println("#"+paramType);

if(paramType instanceof ParameterizedType){

Type[] genericTypes = ((ParameterizedType) paramType).getActualTypeArguments();

for (Type genericType : genericTypes) {

System.out.println("泛型类型："+genericType);

}

}

}

//获得指定方法返回值泛型信息

Method m2 = Demo04.class.getMethod("test02", null);

Type returnType = m2.getGenericReturnType();

if(returnType instanceof ParameterizedType){

Type[] genericTypes = ((ParameterizedType) returnType).getActualTypeArguments();

for (Type genericType : genericTypes) {

System.out.println("返回值，泛型类型："+genericType);

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

package com.bjsxt.test;

import java.lang.annotation.Annotation;

import java.lang.reflect.Field;

import com.bjsxt.test.annotation.SxtField;

import com.bjsxt.test.annotation.SxtTable;

/\*\*

 \* 通过反射获取注解信息

 \* @author 尚学堂高淇

 \*

 \*/

public class Demo05 {

public static void main(String[] args) {

try {

Class clazz = Class.forName("com.bjsxt.test.annotation.SxtStudent");

//获得类的所有有效注解

Annotation[] annotations=clazz.getAnnotations();

for (Annotation a : annotations) {

System.out.println(a);

}

//获得类的指定的注解

SxtTable st = (SxtTable) clazz.getAnnotation(SxtTable.class);

System.out.println(st.value());

//获得类的属性的注解

Field f = clazz.getDeclaredField("studentName");

SxtField sxtField = f.getAnnotation(SxtField.class);

System.out.println(sxtField.columnName()+"--"+sxtField.type()+"--"+sxtField.length());

//根据获得的表名、字段的信息，拼出DDL语句，然后，使用JDBC执行这个SQL，在数据库中生成相关的表

} catch (Exception e) {

e.printStackTrace();

}

}

}

## 反射效率比较，以及设置SetAccessible(true)后的效率对比代码：

## 操作注解（Annotation）的代码如下：

package com.bjsxt.test.annotation;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Target(value={ElementType.FIELD})

@Retention(RetentionPolicy.RUNTIME)

public @interface SxtField {

String columnName();

String type();

int length();

}

package com.bjsxt.test.annotation;

@SxtTable("tb\_student")

public class SxtStudent {

@SxtField(columnName="id",type="int",length=10)

private int id;

@SxtField(columnName="sname",type="varchar",length=10)

private String studentName;

@SxtField(columnName="age",type="int",length=3)

private int age;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getStudentName() {

return studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

package com.bjsxt.test.annotation;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Target(value={ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

public @interface SxtTable {

String value();

}

# 第214集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季214集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 动态编译：

## 实例代码如下：

package com.bjsxt.test;

import java.lang.reflect.Method;

import java.net.URL;

import java.net.URLClassLoader;

import javax.tools.JavaCompiler;

import javax.tools.ToolProvider;

/\*\*

 \* 测试java的动态编译

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo01 {

public static void main(String[] args) throws Exception {

//通过IO流操作，将字符串存储成一个临时文件(Hi.java)，然后调用动态编译方法！

String str = "public class Hi {public static void main(String[] args){System.out.println(\"HaHa,sxt!\");}}";

JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();

int result = compiler.run(null, null, null, "c:/myjava/HelloWorld.java");

System.out.println(result==0?"编译成功":"编译失败");

//通过Runtime调用执行类

// Runtime run = Runtime.getRuntime();

//        Process process = run.exec("java -cp  c:/myjava    HelloWorld");

//

//        InputStream in = process.getInputStream();

//        BufferedReader reader = new BufferedReader(new InputStreamReader(in));

// String info = "";

// while((info=reader.readLine())!=null){

// System.out.println(info);

// }

 try {

             URL[] urls = new URL[] {new URL("file:/"+"C:/myjava/")};

             URLClassLoader loader = new URLClassLoader(urls);

             Class c = loader.loadClass("HelloWorld");

             //调用加载类的main方法

             Method m = c.getMethod("main",String[].class);

             m.invoke(null, (Object)new String[]{});

             //由于可变参数是JDK5.0之后才有。

             //m.invoke(null, (Object)new String[]{});会编译成:m.invoke(null,"aa","bb"),就发生了参数个数不匹配的问题。

             //因此，必须要加上(Object)转型，避免这个问题。

             //public static void main(String[] args)

         } catch (Exception e) {

             e.printStackTrace();

         }

}

}

# 第215集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季215集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 动态调用javaScript代码：

import javax.script.ScriptEngineFactory;

import javax.script.ScriptEngineManager;

/\*\*

 \* 测试java中的脚本引擎调用javascript脚本

 \*

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

}

//执行一个js文件(我们将a.js至于项目的src下即可)

URL is = Demo02.class.getClassLoader().getResource("a.js");

Reader r = new FileReader(is.getPath());

engine.eval(r);

}

}

## js代码如下：

//定义test方法

function test(){

var a = 3;

var b = 4;

println("invoke js file:"+(a+b));

}

//执行test方法

test();

# 第216集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季216集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# java语言的动态性:

    java语言的反射:

    字节码操作：

## javassist入门：

package com.bjsxt.test;

import javassist.CannotCompileException;

import javassist.ClassPool;

import javassist.CtClass;

import javassist.CtConstructor;

import javassist.CtField;

import javassist.CtMethod;

/\*\*

 \* 测试使用javassist生成一个新的类

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo01 {

public static void main(String[] args) throws Exception {

ClassPool pool = ClassPool.getDefault();

CtClass cc = pool.makeClass("com.bjsxt.bean.Emp");

//创建属性

CtField f1 = CtField.make("private int empno;", cc);

CtField f2 = CtField.make("private String ename;", cc);

cc.addField(f1);

cc.addField(f2);

//创建方法

CtMethod m1 = CtMethod.make("public int getEmpno(){return empno;}", cc);

CtMethod m2 = CtMethod.make("public void setEmpno(int empno){this.empno=empno;}", cc);

cc.addMethod(m1);

cc.addMethod(m2);

//添加构造器

CtConstructor constructor = new CtConstructor(new CtClass[]{CtClass.intType,pool.get("java.lang.String")}, cc);

constructor.setBody("{this.empno=empno; this.ename=ename;}");

cc.addConstructor(constructor);

cc.writeFile("c:/myjava"); //将上面构造好的类写入到c:/myjava中

System.out.println("生成类，成功！");

}

}

## emp类：

package com.bjsxt.test;

@Author(name="gaoqi", year=2014)

public class Emp {

private int empno;

private String ename;

public void sayHello(int a){

System.out.println("sayHello,"+a);

}

public int getEmpno() {

return empno;

}

public void setEmpno(int empno) {

this.empno = empno;

}

public String getEname() {

return ename;

}

public void setEname(String ename) {

this.ename = ename;

}

public Emp(int empno, String ename) {

super();

this.empno = empno;

this.ename = ename;

}

public Emp() {

}

}

# 第217集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季217集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Javassist进阶：

## API的用法详解：

package com.bjsxt.test;

import java.lang.reflect.Method;

import java.util.Arrays;

import javassist.ClassPool;

import javassist.CtClass;

import javassist.CtConstructor;

import javassist.CtField;

import javassist.CtMethod;

import javassist.CtNewMethod;

import javassist.Modifier;

import javassist.NotFoundException;

/\*\*

 \* 测试javassist的API

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo02 {

/\*\*

 \* 处理类的基本用法

 \* @throws Exception

 \*/

public static void test01() throws Exception{

ClassPool pool = ClassPool.getDefault();

CtClass cc = pool.get("com.bjsxt.test.Emp");

byte[] bytes = cc.toBytecode();

System.out.println(Arrays.toString(bytes));

System.out.println(cc.getName()); //获取类名

System.out.println(cc.getSimpleName()); //获取简要类名

System.out.println(cc.getSuperclass()); //获得父类

System.out.println(cc.getInterfaces()); //获得接口

}

/\*\*

 \* 测试产生新的方法

 \* @throws Exception

 \*/

public static void test02() throws Exception{

ClassPool pool = ClassPool.getDefault();

CtClass cc = pool.get("com.bjsxt.test.Emp");

// CtMethod m = CtNewMethod.make("public int add(int a,int b){return a+b;}", cc);

CtMethod m = new CtMethod(CtClass.intType,"add",

new CtClass[]{CtClass.intType,CtClass.intType},cc);

m.setModifiers(Modifier.PUBLIC);

m.setBody("{System.out.println(\"www.sxt.cn\");return $1+$2;}");

cc.addMethod(m);

//通过反射调用新生成的方法

Class clazz = cc.toClass();

Object obj = clazz.newInstance();  //通过调用Emp无参构造器，创建新的Emp对象

Method method = clazz.getDeclaredMethod("add", int.class,int.class);

Object result = method.invoke(obj, 200,300);

System.out.println(result);

}

/\*\*

 \* 修改已有的方法的信息，修改方法体的内容

 \* @throws Exception

 \*/

public static void test03() throws Exception{

ClassPool pool = ClassPool.getDefault();

CtClass cc = pool.get("com.bjsxt.test.Emp");

CtMethod cm = cc.getDeclaredMethod("sayHello",new CtClass[]{CtClass.intType});

cm.insertBefore("System.out.println($1);System.out.println(\"start!!!\");");

cm.insertAt(9, "int b=3;System.out.println(\"b=\"+b);");

cm.insertAfter("System.out.println(\"end!!!\");");

//通过反射调用新生成的方法

Class clazz = cc.toClass();

Object obj = clazz.newInstance();  //通过调用Emp无参构造器，创建新的Emp对象

Method method = clazz.getDeclaredMethod("sayHello", int.class);

method.invoke(obj, 300);

}

/\*\*

 \* 属性的操作

 \* @throws Exception

 \*/

public static void test04() throws Exception{

ClassPool pool = ClassPool.getDefault();

CtClass cc = pool.get("com.bjsxt.test.Emp");

// CtField f1 = CtField.make("private int empno;", cc);

CtField f1 = new CtField(CtClass.intType,"salary",cc);

f1.setModifiers(Modifier.PRIVATE);

cc.addField(f1);

// cc.getDeclaredField("ename");   //获取指定的属性

//增加相应的set和get方法

cc.addMethod(CtNewMethod.getter("getSalary", f1));;

cc.addMethod(CtNewMethod.getter("setSalary", f1));;

}

/\*\*

 \* 构造方法的操作

 \* @throws Exception

 \*/

public static void test05() throws Exception {

ClassPool pool = ClassPool.getDefault();

CtClass cc = pool.get("com.bjsxt.test.Emp");

CtConstructor[] cs = cc.getConstructors();

for (CtConstructor c : cs) {

System.out.println(c.getLongName());

}

}

public static void test06() throws Exception{

 CtClass cc = ClassPool.getDefault().get("com.bjsxt.test.Emp");

 Object[] all = cc.getAnnotations();

 Author a = (Author)all[0];

 String name = a.name();

 int year = a.year();

 System.out.println("name: " + name + ", year: " + year);

}

public static void main(String[] args) throws Exception {

test06();

}

}

package com.bjsxt.test;

public @interface Author {

          String name();

           int year();

 }

package com.bjsxt.test;

@Author(name="gaoqi", year=2014)

public class Emp {

private int empno;

private String ename;

public void sayHello(int a){

System.out.println("sayHello,"+a);

}

public int getEmpno() {

return empno;

}

public void setEmpno(int empno) {

this.empno = empno;

}

public String getEname() {

return ename;

}

public void setEname(String ename) {

this.ename = ename;

}

public Emp(int empno, String ename) {

super();

this.empno = empno;

this.ename = ename;

}

public Emp() {

}

}

# 第220集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季220集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 深入类类加载

    引导类加载器（bootstrap class loader);

    扩展类加载器(extensions class loader);

    应用类加载器(application class loader);

自定义类加载器(从java.lang.ClassLoader);

# 双亲委托机制:

目的是保证了java核心库的类型安全：

package com.bjsxt.test;

/\*\*

 \* 测试类加载器的层次结构、双亲委托机制

 \* @author 尚学堂高淇  www.sxt.cn

 \*

 \*/

public class Demo02 {

public static void main(String[] args) {

System.out.println(ClassLoader.getSystemClassLoader());

System.out.println(ClassLoader.getSystemClassLoader().getParent());

System.out.println(ClassLoader.getSystemClassLoader().getParent().getParent());   //JAVA\_HOME/jre/lib/rt.jar

System.out.println(System.getProperty("java.class.path"));

System.out.println("################");

String a = "gaogao";

System.out.println(a.getClass().getClassLoader());

System.out.println(a);

}

}

# 第221集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季221集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 自定义类加载器深入：

## 文件类加载器：

package com.bjsxt.test;

import java.io.ByteArrayOutputStream;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

/\*\*

 \* 自定义文件系统类加载器

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class FileSystemClassLoader extends ClassLoader {

//com.bjsxt.test.User   --> d:/myjava/  com/bjsxt/test/User.class

private String rootDir;

public FileSystemClassLoader(String rootDir){

this.rootDir = rootDir;

}

@Override

protected Class<?> findClass(String name) throws ClassNotFoundException {

Class<?> c = findLoadedClass(name);

//应该要先查询有没有加载过这个类。如果已经加载，则直接返回加载好的类。如果没有，则加载新的类。

if(c!=null){

return c;

}else{

ClassLoader parent = this.getParent();

try {

c = parent.loadClass(name);    //委派给父类加载

} catch (Exception e) {

// e.printStackTrace();

}

if(c!=null){

return c;

}else{

byte[] classData = getClassData(name);

if(classData==null){

throw new ClassNotFoundException();

}else{

c = defineClass(name, classData, 0,classData.length);

}

}

}

return c;

}

private byte[] getClassData(String classname){   //com.bjsxt.test.User   d:/myjava/  com/bjsxt/test/User.class

String path = rootDir +"/"+ classname.replace('.', '/')+".class";

// IOUtils,可以使用它将流中的数据转成字节数组

InputStream is = null;

ByteArrayOutputStream baos = new ByteArrayOutputStream();

try{

is  = new FileInputStream(path);

byte[] buffer = new byte[1024];

int temp=0;

while((temp=is.read(buffer))!=-1){

baos.write(buffer, 0, temp);

}

return baos.toByteArray();

}catch(Exception e){

e.printStackTrace();

return null;

}finally{

try {

if(is!=null){

is.close();

}

} catch (IOException e) {

e.printStackTrace();

}

try {

if(baos!=null){

baos.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

## 测试类的代码如下：

package com.bjsxt.test;

/\*\*

 \* 测试自定义的FileSystemClassLoader

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo03 {

public static void main(String[] args) throws Exception{

FileSystemClassLoader loader = new FileSystemClassLoader("d:/myjava");

FileSystemClassLoader loader2 = new FileSystemClassLoader("d:/myjava");

Class<?> c = loader.loadClass("com.bjsxt.gaoqi.HelloWorld");

Class<?> c2 = loader.loadClass("com.bjsxt.gaoqi.HelloWorld");

Class<?> c3 = loader2.loadClass("com.bjsxt.gaoqi.HelloWorld");

Class<?> c4 = loader2.loadClass("java.lang.String");

Class<?> c5 = loader2.loadClass("com.bjsxt.test.Demo01");

System.out.println(c.hashCode());

System.out.println(c2.hashCode());

System.out.println(c3.hashCode()); //同一个类，被不同的加载器加载，JVM认为也是不相同的类

System.out.println(c4.hashCode());

System.out.println(c4.getClassLoader()); //引导类加载器

System.out.println(c3.getClassLoader()); //自定义的类加载器

System.out.println(c5.getClassLoader()); //系统默认的类加载器

}

}

## 网络类加载器：

package com.bjsxt.test;

import java.io.ByteArrayOutputStream;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.net.URL;

/\*\*

 \* 网络类加载器

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class NetClassLoader extends ClassLoader {

//com.bjsxt.test.User   --> www.sxt.cn/myjava/  com/bjsxt/test/User.class

private String rootUrl;

public NetClassLoader(String rootUrl){

this.rootUrl = rootUrl;

}

@Override

protected Class<?> findClass(String name) throws ClassNotFoundException {

Class<?> c = findLoadedClass(name);

//应该要先查询有没有加载过这个类。如果已经加载，则直接返回加载好的类。如果没有，则加载新的类。

if(c!=null){

return c;

}else{

ClassLoader parent = this.getParent();

try {

c = parent.loadClass(name);    //委派给父类加载

} catch (Exception e) {

// e.printStackTrace();

}

if(c!=null){

return c;

}else{

byte[] classData = getClassData(name);

if(classData==null){

throw new ClassNotFoundException();

}else{

c = defineClass(name, classData, 0,classData.length);

}

}

}

return c;

}

private byte[] getClassData(String classname){   //com.bjsxt.test.User   d:/myjava/  com/bjsxt/test/User.class

String path = rootUrl +"/"+ classname.replace('.', '/')+".class";

// IOUtils,可以使用它将流中的数据转成字节数组

InputStream is = null;

ByteArrayOutputStream baos = new ByteArrayOutputStream();

try{

URL url = new URL(path);

is  = url.openStream();

byte[] buffer = new byte[1024];

int temp=0;

while((temp=is.read(buffer))!=-1){

baos.write(buffer, 0, temp);

}

return baos.toByteArray();

}catch(Exception e){

e.printStackTrace();

return null;

}finally{

try {

if(is!=null){

is.close();

}

} catch (IOException e) {

e.printStackTrace();

}

try {

if(baos!=null){

baos.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

# 第222集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季222集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 类加载器之加密技术初步：

## 自定义加密解密类加载器技术展示的代码如下：

package com.bjsxt.test;

/\*\*

 \* 测试简单加密解密(取反)操作

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo04 {

public static void main(String[] args) throws Exception {

//测试取反操作

// int a = 3; //0000011

// System.out.println(Integer.toBinaryString(a^0xff));

//加密后的class文件，正常的类加载器无法加载，报classFormatError

// FileSystemClassLoader loader = new FileSystemClassLoader("d:/myjava/temp");

// Class<?> c = loader.loadClass("HelloWorld");

// System.out.println(c);

DecrptClassLoader loader = new DecrptClassLoader("d:/myjava/temp");

Class<?> c = loader.loadClass("HelloWorld");

System.out.println(c);

}

}

## 加密类加载器代码：

package com.bjsxt.test;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

/\*\*

 \* 加密工具类

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class EncrptUtil {

public static void main(String[] args) {

encrpt("d:/myjava/HelloWorld.class", "d:/myjava/temp/HelloWorld.class");

}

public static void encrpt(String src, String dest){

FileInputStream fis = null;

FileOutputStream fos = null;

try {

fis = new FileInputStream(src);

fos = new FileOutputStream(dest);

int temp = -1;

while((temp=fis.read())!=-1){

fos.write(temp^0xff);  //取反操作

}

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(fis!=null){

fis.close();

}

} catch (IOException e) {

e.printStackTrace();

}

try {

if(fos!=null){

fos.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

## 解密的类加载器代码如下：

package com.bjsxt.test;

import java.io.ByteArrayOutputStream;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

/\*\*

 \* 加载文件系统中加密后的class字节码的类加载器

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class DecrptClassLoader  extends ClassLoader {

 //com.bjsxt.test.User   --> d:/myjava/  com/bjsxt/test/User.class

 private String rootDir;

 public DecrptClassLoader(String rootDir){

 this.rootDir = rootDir;

 }

 return c;

 }

 private byte[] getClassData(String classname){   //com.bjsxt.test.User   d:/myjava/  com/bjsxt/test/User.class

 String path = rootDir +"/"+ classname.replace('.', '/')+".class";

// IOUtils,可以使用它将流中的数据转成字节数组

 InputStream is = null;

 ByteArrayOutputStream baos = new ByteArrayOutputStream();

 try{

 is  = new FileInputStream(path);

 int temp = -1;

 while((temp=is.read())!=-1){

 baos.write(temp^0xff);  //取反操作,相当于解密

 }

 return baos.toByteArray();

 }catch(Exception e){

 e.printStackTrace();

 return null;

 }finally{

 try {

 if(is!=null){

 is.close();

 }

 } catch (IOException e) {

 e.printStackTrace();

 }

 try {

 if(baos!=null){

 baos.close();

 }

 } catch (IOException e) {

 e.printStackTrace();

 }

 }

 }

}

# 第223集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季223集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 线程上下文类加载器：

package com.bjsxt.test;

/\*\*

\* 线程上下文类加载器的测试

\* @author 尚学堂高淇 www.sxt.cn

\*

\*/

public class Demo05 {

public static void main(String[] args) throws Exception {

ClassLoader loader = Demo05.class.getClassLoader();

System.out.println(loader);

ClassLoader loader2 = Thread.currentThread().getContextClassLoader();

System.out.println(loader2);

Thread.currentThread().setContextClassLoader(new FileSystemClassLoader("d:/myjava/"));

System.out.println(Thread.currentThread().getContextClassLoader());

Class<Demo01> c = (Class<Demo01>) Thread.currentThread().getContextClassLoader().loadClass("com.bjsxt.test.Demo01");

System.out.println(c);

System.out.println(c.getClassLoader());

}

}

其余的扩展可以看视频增加了解即可了

# 

# 第224集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季224集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 内部类技术

    Nested Class 内部类，也称为嵌套类。

    静态内部类：

    成员内部类：

## 内部类的存在意义：

        1）方便使用外部类的属性

        2）内部类的存在就是为了辅助外部类实现的功能的。

package com.bjsxs.test;

/\*\*

 \* 测试内部类

 \* @author 尚学堂高淇www.bjsxt.cn

 \*

 \*/

public class Demo01 {

//静态内部类

public static class StaticNestedClass{

}

//普通内部类(成员内部类)

private static class FieldInnerClass{

}

void sayHello(){

//方法内部类

class LocalClass{

}

//匿名内部类

Runnable runnable = new Runnable(){//两个作用：定义了匿名内部类的类体，创建了内部类的一个实例。

@Override

public void run() {

System.out.println("我是匿名内部类");

}

};

}

}

package com.bjsxs.test;

import com.bjsxs.test.Outer02.StaticInnerClass;

/\*\*

 \* 测试静态内部类的详细用法

 \*

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo02 {

public static void main(String[] args) {

// Outer02.StaticInnerClass sic = new Outer02.StaticInnerClass();

StaticInnerClass sic2 = new StaticInnerClass();

}

}

class Outer02 {

int c = 5;

static int d = 10;

void ttt() {

StaticInnerClass sic = new StaticInnerClass();

}

// 静态内部类

public static class StaticInnerClass {

int a = 3;

static int b = 5;

public void test() {

System.out.println(d);

// System.out.println(c);//静态内部类不能调用外部类的普通成员，只能调用静态成员。

}

}

}

package com.bjsxs.test;

import java.util.MissingFormatArgumentException;

/\*\*

 \* 内部类的规则的测试如下

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo03 {

public static void main(String[] args) {

Outer03 out = new Outer03();

Outer03.InnerClass ic = out.new InnerClass();

ic.test();

}

}

class Outer03 {

private int a = 3;

int b = 10;

public void ttt() {

InnerClass ic = new InnerClass();

}

// 成员内部类

class InnerClass {

int c = 1;

// static int

// d=80;//成员内部类不能有静态的成员，或者static方法，除非声明为final的。并且只能是编译器可以确定值的常量表达式。

// final static Date d2=new InnerClass();

// static InnerClass ab =new

// InnerClass();//Outer03.InnerClass.ab可以直接获得内部类对象，这时成员内部类对象就完全脱离了外部类的控制，与设计成语内部类的初衷不符。

void test() {

System.out.println(a);// 1内部可以方便的使用外部类的成员，2，内部的存在就是为了辅助外部类实现功能。

System.out.println("内部类的对象" + this);

System.out.println("外部类对象" + Outer03.this);

}

}

}

# 第225集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季225集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 方法内部类：

package com.bjsxs.test;

/\*\*

 \* 测试方法内部类（局部内部类）

 \*

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Demo04 {

public static void main(String[] args) {

}

}

class Outer04 {

public void test() {

final int a = 3;

class Inner {// 方法内部类只在本方法内可见！

int b=10;

//static int c=20;//方法内部类不能定义静态成员。

void tt(){

System.out.println(b);

System.out.println(a);//方法内部类不能够引用所在方法的普通局部变量，除非是常量！

//原因如下：

//方法的生命周期和方法内部类对象的生命周期不一致！

}

}

Inner i = new Inner(){

};

}

void test2() {

}

}

# 匿名内部类:

package com.bjsxs.test;

/\*\*

 \* 匿名内部类的声明方式： 继承式 ，参数式，接口式

 \*

 \* @author Administrator

 \*

 \*/

public class Demo05 {

public static void main(String[] args) {

Outer05 out = new Outer05();

out.test();

}

}

class Outer05 {

public void test02(Car c) {

c.run();

}

public void test() {

final int a = 3;

// 匿名内部类，由于本内部类定义在方法中，同时也是方法内部类

Runnable runnable = new Runnable() {

@Override

public void run() {

}

};

// 匿名内部类的，继承式写法

Car car = new Car() {

@Override

public void run() {

System.out.println("子类的车跑了！");

System.out.println(a);

}

};

car.run();

test02(new Car() {

@Override

public void run() {

System.out.println("参数式匿名内部类，车再跑！");

}

});

}

}

class Car {

public void run() {

System.out.println("汽车跑！");

}

}

# 第226集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季226集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# GOF23设计模式（group of four ）

    单例模式：

            保证一个类只能产生一个对象！

    饿汉式（线程安全，调用效率高，但是不能延迟加载）

package com.bjsxt.singleton;

/\*\*

 \* 测试饿汉式单例模式

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class SingletonDemo1 {

//类初始化时，立即加载这个对象（没有延时加载的优势）。加载类时，天然的是线程安全的！

private static SingletonDemo1 instance = new SingletonDemo1();

private SingletonDemo1(){

}

//方法没有同步，调用效率高！

public static SingletonDemo1  getInstance(){

return instance;

}

}

    懒汉式（线程安全，调用效率不高，但是可以延迟加载）

package com.bjsxt.singleton;

/\*\*

 \* 测试懒汉式单例模式

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class SingletonDemo2 {

//类初始化时，不初始化这个对象（延时加载，真正用的时候再创建）。

private static SingletonDemo2 instance;

private SingletonDemo2(){ //私有化构造器

}

//方法同步，调用效率低！

public static  synchronized SingletonDemo2  getInstance(){

if(instance==null){

instance = new SingletonDemo2();

}

return instance;

}

}

# 第227集

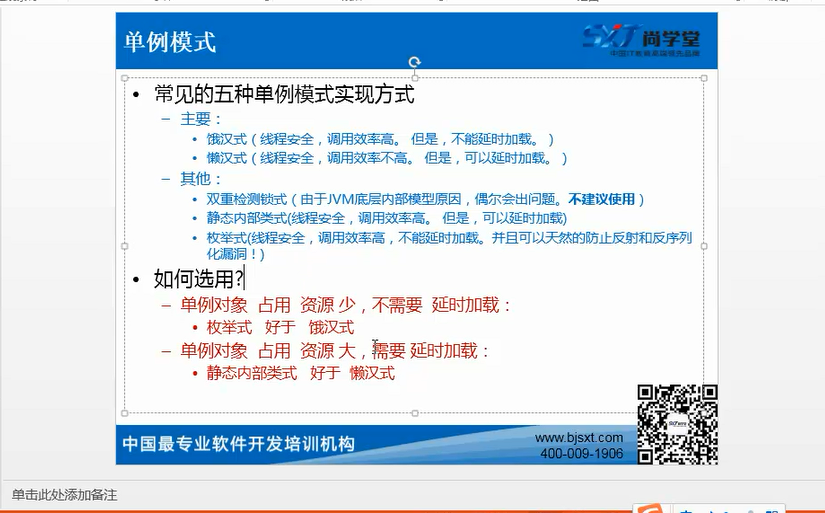
尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季227集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)



# 单例模式之双重检测锁实现：

package com.bjsxt.singleton;

/\*\*

 \* 双重检查锁实现单例模式

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class SingletonDemo3 {

  private static SingletonDemo3 instance = null;

  public static SingletonDemo3 getInstance() {

    if (instance == null) {

      SingletonDemo3 sc;

      synchronized (SingletonDemo3.class) {

        sc = instance;

        if (sc == null) {

          synchronized (SingletonDemo3.class) {

            if(sc == null) {

              sc = new SingletonDemo3();

            }

          }

          instance = sc;

        }

      }

    }

    return instance;

  }

  private SingletonDemo3() {

  }

}

# 静态内部类实现单例模式：

package com.bjsxt.singleton;

/\*\*

 \* 测试静态内部类实现单例模式

 \* 这种方式：线程安全，调用效率高，并且实现了延时加载！

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class SingletonDemo4 {

private static class SingletonClassInstance {

private static final SingletonDemo4 instance = new SingletonDemo4();

}

private SingletonDemo4(){

}

//方法没有同步，调用效率高！

public static SingletonDemo4  getInstance(){

return SingletonClassInstance.instance;

}

}

# 枚举方式实现单例模式：

package com.bjsxt.singleton;

/\*\*

 \* 测试枚举式实现单例模式(没有延时加载)

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public enum SingletonDemo5 {

//这个枚举元素，本身就是单例对象！

INSTANCE;

//添加自己需要的操作！

public void singletonOperation(){

}

}

# 第228集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季228集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 单例模式如何防止反射和反序列化的漏洞！

package com.bjsxt.singleton;

import java.io.ObjectStreamException;

import java.io.Serializable;

/\*\*

 \* 测试懒汉式单例模式(如何防止反射和反序列化漏洞)

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class SingletonDemo6 implements Serializable {

//类初始化时，不初始化这个对象（延时加载，真正用的时候再创建）。

private static SingletonDemo6 instance;

private SingletonDemo6(){ //私有化构造器

if(instance!=null){

throw new RuntimeException();

}

}

//方法同步，调用效率低！

public static  synchronized SingletonDemo6  getInstance(){

if(instance==null){

instance = new SingletonDemo6();

}

return instance;

}

//反序列化时，如果定义了readResolve()则直接返回此方法指定的对象。而不需要单独再创建新对象！

private Object readResolve() throws ObjectStreamException {

return instance;

}

}

## 测试反射和反序列化的漏洞

package com.bjsxt.singleton;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutput;

import java.io.ObjectOutputStream;

import java.lang.reflect.Constructor;

/\*\*

 \* 测试反射和反序列化破解单例模式

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Client2 {

public static void main(String[] args) throws Exception {

SingletonDemo6 s1 = SingletonDemo6.getInstance();

SingletonDemo6 s2 = SingletonDemo6.getInstance();

System.out.println(s1);

System.out.println(s2);

//通过反射的方式直接调用私有构造器

// Class<SingletonDemo6> clazz = (Class<SingletonDemo6>) Class.forName("com.bjsxt.singleton.SingletonDemo6");

// Constructor<SingletonDemo6> c = clazz.getDeclaredConstructor(null);

// c.setAccessible(true);

// SingletonDemo6  s3 = c.newInstance();

// SingletonDemo6  s4 = c.newInstance();

// System.out.println(s3);

// System.out.println(s4);

//通过反序列化的方式构造多个对象

FileOutputStream fos = new FileOutputStream("d:/a.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(s1);

oos.close();

fos.close();

ObjectInputStream ois = new ObjectInputStream(new FileInputStream("d:/a.txt"));

SingletonDemo6 s3 =  (SingletonDemo6) ois.readObject();

System.out.println(s3);

}

}

## 测试单例的5种情况的效率：

package com.bjsxt.singleton;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutput;

import java.io.ObjectOutputStream;

import java.lang.reflect.Constructor;

import java.util.concurrent.CountDownLatch;

/\*\*

 \* 测试多线程环境下五种创建单例模式的效率

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Client3 {

public static void main(String[] args) throws Exception {

long start = System.currentTimeMillis();

int threadNum = 10;

final CountDownLatch  countDownLatch = new CountDownLatch(threadNum);

for(int i=0;i<threadNum;i++){

new Thread(new Runnable() {

@Override

public void run() {

for(int i=0;i<1000000;i++){

// Object o = SingletonDemo4.getInstance();

Object o = SingletonDemo5.INSTANCE;

}

countDownLatch.countDown();

}

}).start();

}

countDownLatch.await(); //main线程阻塞，直到计数器变为0，才会继续往下执行！

long end = System.currentTimeMillis();

System.out.println("总耗时："+(end-start));

}

}

# 第229集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季229集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 工厂模式

    oop设计原则：

    ocp:对源码关闭，对扩展开发原则

    dip:依赖翻转原则。

     迪米特法则：

    定义一个接口Car

package com.bjsxt.factory.simplefactory;

public interface Car {

void run();

}

## 实现类Audi

package com.bjsxt.factory.simplefactory;

public class Audi implements Car {

@Override

public void run() {

System.out.println("奥迪再跑！");

}

}

## 实现类Byd

package com.bjsxt.factory.simplefactory;

public class Byd implements Car {

@Override

public void run() {

System.out.println("比亚迪再跑！");

}

}

## 测试类Client01,没有工厂模式的时候：

package com.bjsxt.factory.simplefactory;

/\*\*

 \* 测试在没有工厂模式的情况下

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Client01 {   //调用者

 public static void main(String[] args) {

 Car c1 = new Audi();

 Car c2 = new Byd();

 c1.run();

 c2.run();

 }

}

## 添加一个工厂类CarFactory

package com.bjsxt.factory.simplefactory;

public class CarFactory {

public static  Car createCar(String type){

if("奥迪".equals(type)){

return new Audi();

}else if("比亚迪".equals(type)){

return new Byd();

}else{

return null;

}

}

}

## 工厂类也可以这样

package com.bjsxt.factory.simplefactory;

/\*\*

 \* 简单工厂类

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class CarFactory2 {

public static  Car createAudi(){

return new Audi();

}

public static  Car createByd(){

return new Byd();

}

}

## 测试类

package com.bjsxt.factory.simplefactory;

/\*\*

 \* 简单工厂情况下

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Client02 {   //调用者

public static void main(String[] args) {

Car c1 =CarFactory.createCar("奥迪");

Car c2 = CarFactory.createCar("比亚迪");

c1.run();

c2.run();

}

}

# 第230集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季230集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 工厂模式

    --实现了创建者和调用者的分离。

   --详细分类：

        -简单工厂模式

        -工厂方法模式

        -抽象工厂模式

    --面向对象设计的基本原则：

    -OCP（开闭原则，Open-Closed Principle）:一个软件的实体应当对扩展开放，对修改关闭。

    -DIP（依赖倒转原则，Dependence Inversion Principle）要针对接口编程，不要针对实现编程。

    -LOD（迪米特法则，Law of Demeter）:只与你直接的朋友通信，而避免与陌生人通信。

    工厂方法的代码如下：

## 汽车接口：

package com.bjsxt.factory.factorymethod;

public interface Car {

void run();

}

## 汽车工厂接口：

package com.bjsxt.factory.factorymethod;

public class Audi implements Car {

@Override

public void run() {

System.out.println("奥迪再跑！");

}

}

## 车的实现类如下：

package com.bjsxt.factory.factorymethod;

public class Audi implements Car { @Override

public void run() {

System.out.println("奥迪再跑！");

}

}

package com.bjsxt.factory.factorymethod;

public class Byd implements Car {

@Override

public void run() {

System.out.println("比亚迪再跑！");

}

}

package com.bjsxt.factory.factorymethod;

public class AudiFactory implements CarFactory {

@Override

public Car createCar() {

return new Audi();

}

}

package com.bjsxt.factory.factorymethod;

public class BydFactory implements CarFactory {

@Override

public Car createCar() {

return new Byd();

}

}

## 测试类

package com.bjsxt.factory.factorymethod;

public class Client {

public static void main(String[] args) {

Car c1 = new AudiFactory().createCar();

Car c2 = new BydFactory().createCar();

c1.run();

c2.run();

}

}

## 增加一个实现类

package com.bjsxt.factory.factorymethod;

public class Benz implements Car {

@Override

public void run() {

System.out.println("奔驰再跑！");

}

}

package com.bjsxt.factory.factorymethod;

public class BenzFactory implements CarFactory {

@Override

public Car createCar() {

return new Benz();

}

}

# 简单工厂，工厂方法：

    1)结构复杂度：简单工厂模式优胜。

    2）代码复杂度：简单工厂模式占优。

    3）客户端复杂度：

    来体会一下两个模式的优劣度！

    根据设计理论建议：工厂方法模式，但是实际上，我们一般都是使用简单工厂模式！

# 第231集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季231集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 抽象工厂模式：

多个接口，构成了产品族的时候使用抽象工厂模式！

package com.bjsxt.factory.abstractFactory;

public interface Engine {

void run();

void start();

}

class LuxuryEngine implements Engine{

@Override

public void run() {

System.out.println("转的快！");

}

@Override

public void start() {

System.out.println("启动快!可以自动启停！");

}

}

class LowEngine implements Engine{

@Override

public void run() {

System.out.println("转的慢！");

}

@Override

public void start() {

System.out.println("启动慢!");

}

}

## 接口

package com.bjsxt.factory.abstractFactory;

public interface Seat {

void massage();

}

class LuxurySeat implements Seat {

@Override

public void massage() {

System.out.println("可以自动按摩！");

}

}

class LowSeat implements Seat {

@Override

public void massage() {

System.out.println("不能按摩！");

}

}

package com.bjsxt.factory.abstractFactory;

public interface Tyre {

void revolve();

}

class LuxuryTyre implements Tyre {

@Override

public void revolve() {

System.out.println("旋转不磨损！");

}

}

class LowTyre implements Tyre {

@Override

public void revolve() {

System.out.println("旋转磨损快！");

}

}

## 汽车工厂接口：

package com.bjsxt.factory.abstractFactory;

public interface CarFactory {

Engine createEngine();

Seat createSeat();

Tyre createTyre();

}

package com.bjsxt.factory.abstractFactory;

public class LuxuryCarFactory implements CarFactory {

@Override

public Engine createEngine() {

return new LuxuryEngine();

}

@Override

public Seat createSeat() {

return new LuxurySeat();

}

@Override

public Tyre createTyre() {

return new LuxuryTyre();

}

}

package com.bjsxt.factory.abstractFactory;

public class LowCarFactory implements CarFactory {

@Override

public Engine createEngine() {

return new LowEngine();

}

@Override

public Seat createSeat() {

return new LowSeat();

}

@Override

public Tyre createTyre() {

return new LowTyre();

}

}

## 测试代码：

package com.bjsxt.factory.abstractFactory;

public class Client {

public static void main(String[] args) {

CarFactory  factory = new LuxuryCarFactory();

Engine e = factory.createEngine();

e.run();

e.start();

}

}

# 第232集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季232集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 建造者模式：

package com.bjsxt.builder;

/\*\*

 \* 尚学堂牌宇宙飞船

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class AirShip {

private OrbitalModule orbitalModule;  //轨道舱

private Engine engine; //发动机

private EscapeTower escapeTower;  //逃逸塔

public void launch(){

System.out.println("发射！");

}

public OrbitalModule getOrbitalModule() {

return orbitalModule;

}

public void setOrbitalModule(OrbitalModule orbitalModule) {

this.orbitalModule = orbitalModule;

}

public Engine getEngine() {

return engine;

}

public void setEngine(Engine engine) {

this.engine = engine;

}

public EscapeTower getEscapeTower() {

return escapeTower;

}

public void setEscapeTower(EscapeTower escapeTower) {

this.escapeTower = escapeTower;

}

}

class OrbitalModule{

private String name;

public OrbitalModule(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

class Engine {

private String name;

public Engine(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

class EscapeTower{

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public EscapeTower(String name) {

super();

this.name = name;

}

}

package com.bjsxt.builder;

public interface AirShipBuilder {

Engine builderEngine();

OrbitalModule builderOrbitalModule();

EscapeTower  builderEscapeTower();

}

package com.bjsxt.builder;

public interface AirShipDirector {

/\*\*

 \* 组装飞船对象

 \* @return

 \*/

AirShip   directAirShip();

}

package com.bjsxt.builder;

public class SxtAirShipBuilder implements AirShipBuilder {

//StringBuilder, 以后学习XML解析中，JDOM库中的类：DomBuilder,SaxBuilder

@Override

public Engine builderEngine() {

System.out.println("构建尚学堂牌发动机！");

return new Engine("尚学堂牌发动机！");

}

@Override

public EscapeTower builderEscapeTower() {

System.out.println("构建逃逸塔");

return new EscapeTower("尚学堂牌逃逸塔");

}

@Override

public OrbitalModule builderOrbitalModule() {

System.out.println("构建轨道舱");

return new OrbitalModule("尚学堂牌轨道舱");

}

}

package com.bjsxt.builder;

public class SxtAirshipDirector implements AirShipDirector {

private AirShipBuilder builder;

public SxtAirshipDirector(AirShipBuilder builder) {

this.builder = builder;

}

@Override

public AirShip directAirShip() {

Engine e = builder.builderEngine();

OrbitalModule o = builder.builderOrbitalModule();

EscapeTower et = builder.builderEscapeTower();

//装配成飞船对象

AirShip ship = new AirShip();

ship.setEngine(e);

ship.setEscapeTower(et);

ship.setOrbitalModule(o);

return ship;

}

}

以上代码是对应视频的代码，供参考！构建中模式的思想可以看视频进一步理解。

# 第233-234集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季233-234集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 原型模式

    当使用New关键字实例化对象的时候，很复杂，可以考虑原型模式来实现！

    再则就是New对象比较耗时,克隆（clone）比较节省时间！

## 浅克隆：

package com.bjsxt.prototype;

import java.io.Serializable;

import java.util.Date;

public class Sheep implements Cloneable,Serializable {   //1997,英国的克隆羊，多利！

private String sname;

private Date birthday;

@Override

protected Object clone() throws CloneNotSupportedException {

Object obj = super.clone();  //直接调用object对象的clone()方法！

return obj;

}

public String getSname() {

return sname;

}

public void setSname(String sname) {

this.sname = sname;

}

public Date getBirthday() {

return birthday;

}

public void setBirthday(Date birthday) {

this.birthday = birthday;

}

public Sheep(String sname, Date birthday) {

super();

this.sname = sname;

this.birthday = birthday;

}

public Sheep() {

}

}

package com.bjsxt.prototype;

import java.util.Date;

/\*\*

 \* 测试原型模式(浅克隆)

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Client {

public static void main(String[] args) throws Exception {

Date date = new Date(12312321331L);

Sheep s1 = new Sheep("少利",date);

System.out.println(s1);

System.out.println(s1.getSname());

System.out.println(s1.getBirthday());

date.setTime(23432432423L);

System.out.println(s1.getBirthday());

Sheep s2 = (Sheep) s1.clone();

s2.setSname("多利");

System.out.println(s2);

System.out.println(s2.getSname());

System.out.println(s2.getBirthday());

}

}

## 深克隆

package com.bjsxt.prototype;

import java.util.Date;

//测试深复制

public class Sheep2 implements Cloneable {   //1997,英国的克隆羊，多利！

private String sname;

private Date birthday;

@Override

protected Object clone() throws CloneNotSupportedException {

Object obj = super.clone();  //直接调用object对象的clone()方法！

//添加如下代码实现深复制(deep Clone)

Sheep2 s = (Sheep2) obj;

s.birthday = (Date) this.birthday.clone();  //把属性也进行克隆！

return obj;

}

public String getSname() {

return sname;

}

public void setSname(String sname) {

this.sname = sname;

}

public Date getBirthday() {

return birthday;

}

public void setBirthday(Date birthday) {

this.birthday = birthday;

}

public Sheep2(String sname, Date birthday) {

super();

this.sname = sname;

this.birthday = birthday;

}

public Sheep2() {

}

}

### 测试：

package com.bjsxt.prototype;

import java.util.Date;

/\*\*

 \* 原型模式(深复制)

 \* @author 尚学堂高淇www.sxt.cn

 \*

 \*/

public class Client2 {

public static void main(String[] args) throws CloneNotSupportedException {

Date date = new Date(12312321331L);

Sheep2 s1 = new Sheep2("少利",date);

Sheep2 s2 = (Sheep2) s1.clone();   //实现深复制。s2对象的birthday是一个新对象！

System.out.println(s1);

System.out.println(s1.getSname());

System.out.println(s1.getBirthday());

date.setTime(23432432423L);

System.out.println(s1.getBirthday());

s2.setSname("多利");

System.out.println(s2);

System.out.println(s2.getSname());

System.out.println(s2.getBirthday());

}

}

## 深克隆的另一种实现方式，序列化和反序列化实现身：

package com.bjsxt.prototype;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.Date;

/\*\*

 \* 原型模式(深复制,使用序列化和反序列化的方式实现深复制)

 \* @author 尚学堂高淇www.sxt.cn

 \*

 \*/

public class Client3 {

public static void main(String[] args) throws CloneNotSupportedException, Exception {

Date date = new Date(12312321331L);

Sheep s1 = new Sheep("少利",date);

System.out.println(s1);

System.out.println(s1.getSname());

System.out.println(s1.getBirthday());

// 使用序列化和反序列化实现深复制

ByteArrayOutputStream bos = new ByteArrayOutputStream();

ObjectOutputStream    oos = new ObjectOutputStream(bos);

oos.writeObject(s1);

byte[] bytes = bos.toByteArray();

ByteArrayInputStream  bis = new ByteArrayInputStream(bytes);

ObjectInputStream   ois = new ObjectInputStream(bis);

Sheep s2 = (Sheep) ois.readObject();   //克隆好的对象！

System.out.println("修改原型对象的属性值");

date.setTime(23432432423L);

System.out.println(s1.getBirthday());

s2.setSname("多利");

System.out.println(s2);

System.out.println(s2.getSname());

System.out.println(s2.getBirthday());

}

}

### 测试类

package com.bjsxt.prototype;

/\*\*

 \* 测试普通new方式创建对象和clone方式创建对象的效率差异！

 \* 如果需要短时间创建大量对象，并且new的过程比较耗时。则可以考虑使用原型模式！

 \* @author 尚学堂高淇 www.sxt.cn

 \*

 \*/

public class Client4 {

public static void testNew(int size){

long start = System.currentTimeMillis();

for(int i=0;i<size;i++){

Laptop t = new Laptop();

}

long end = System.currentTimeMillis();

System.out.println("new的方式创建耗时："+(end-start));

}

public static void testClone(int size) throws CloneNotSupportedException{

long start = System.currentTimeMillis();

Laptop t = new Laptop();

for(int i=0;i<size;i++){

Laptop temp = (Laptop) t.clone();

}

long end = System.currentTimeMillis();

System.out.println("clone的方式创建耗时："+(end-start));

}

public static void main(String[] args) throws Exception {

testNew(1000);

testClone(1000);

}

}

class Laptop implements Cloneable {  //笔记本电脑

public Laptop() {

try {

Thread.sleep(10);  //模拟创建对象耗时的过程!

} catch (InterruptedException e) {

e.printStackTrace();

}

}

@Override

protected Object clone() throws CloneNotSupportedException {

Object obj = super.clone();  //直接调用object对象的clone()方法！

return obj;

}

}

# 第235集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季235集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 适配器模式

     对象适配器，类适配器模式！

package com.bjsxt.adapter;

/\*\*

 \* 被适配的类

 \* (相当于例子中的，PS/2键盘)

 \* @author Administrator

 \*

 \*/

public class Adaptee {

public void request(){

System.out.println("可以完成客户请求的需要的功能！");

}

}

package com.bjsxt.adapter;

/\*\*

 \* 适配器 (类适配器方式)

 \* (相当于usb和ps/2的转接器)

 \*/

public class Adapter extends Adaptee implements Target {

@Override

public void handleReq() {

super.request();

}

}

package com.bjsxt.adapter;

public interface Target {

void handleReq();

}

package com.bjsxt.adapter;

/\*\*

 \* 客户端类

 \* (相当于例子中的笔记本，只有USB接口)

 \* @author Administrator

 \*

 \*/

public class Client {

public void test1(Target t){

t.handleReq();

}

public static void main(String[] args) {

Client  c = new Client();

Adaptee a = new Adaptee();

// Target t = new Adapter();

Target t = new Adapter2(a);

c.test1(t);

}

}

package com.bjsxt.adapter;

/\*\*

 \* 适配器 (对象适配器方式,使用了组合的方式跟被适配对象整合)

 \* (相当于usb和ps/2的转接器)

 \* @author Administrator

 \*

 \*/

public class Adapter2  implements Target {

private Adaptee adaptee;

@Override

public void handleReq() {

adaptee.request();

}

public Adapter2(Adaptee adaptee) {

super();

this.adaptee = adaptee;

}

}

# 第238集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季238集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 桥接模式：

    单一职责的原则的下：

    作用：

            1）处理多层继承结构，处理多维度变化的场景，将各个维度设计成独立的继承结构，使各个维度可以独立的扩展在抽象层建立关联。

package com.bjsxt.bridge;

/\*\*

 \* 品牌

 \* @author Administrator

 \*

 \*/

public interface Brand {

void sale();

}

class Lenovo implements Brand {

@Override

public void sale() {

System.out.println("销售联想电脑");

}

}

class Dell implements Brand {

@Override

public void sale() {

System.out.println("销售Dell电脑");

}

}

class Shenzhou implements Brand {

@Override

public void sale() {

System.out.println("销售神舟电脑");

}

}

## 商品类

package com.bjsxt.bridge;

public interface Computer {

void sale();

}

class Desktop implements Computer {

@Override

public void sale() {

System.out.println("销售台式机！");

}

}

class Laptop implements Computer {

@Override

public void sale() {

System.out.println("销售笔记本！");

}

}

class Pad implements Computer {

@Override

public void sale() {

System.out.println("销售平板电脑！");

}

}

class LenovoDesktop extends Desktop {

@Override

public void sale() {

System.out.println("销售联想台式机");

}

}

class LenovoLaptop extends Laptop {

@Override

public void sale() {

System.out.println("销售联想笔记本");

}

}

class LenovoPad extends Pad {

@Override

public void sale() {

System.out.println("销售联想平板电脑");

}

}

class ShenzhouDesktop extends Desktop {

@Override

public void sale() {

System.out.println("销售神舟台式机");

}

}

class ShenzhouLaptop extends Laptop {

@Override

public void sale() {

System.out.println("销售神舟笔记本");

}

}

class ShenzhouPad extends Pad {

@Override

public void sale() {

System.out.println("销售神舟平板电脑");

}

}

class DellDesktop extends Desktop {

@Override

public void sale() {

System.out.println("销售戴尔台式机");

}

}

class DellLaptop extends Laptop {

@Override

public void sale() {

System.out.println("销售戴尔笔记本");

}

}

class DellPad extends Pad {

@Override

public void sale() {

System.out.println("销售戴尔平板电脑");

}

}

### 测试：

package com.bjsxt.bridge;

/\*\*

 \* 电脑类型的维度

 \* @author Administrator

 \*

 \*/

public class Computer2 {

protected Brand brand;

public Computer2(Brand b) {

this.brand = b;

}

public void sale(){

brand.sale();

}

}

class Desktop2 extends Computer2 {

public Desktop2(Brand b) {

super(b);

}

@Override

public void sale() {

super.sale();

System.out.println("销售台式机");

}

}

class Laptop2 extends Computer2 {

public Laptop2(Brand b) {

super(b);

}

@Override

public void sale() {

super.sale();

System.out.println("销售笔记本");

}

}

# 使用桥接模式后的测试代码如下：

package com.bjsxt.bridge;

public class Client {

public static void main(String[] args) {

//销售联想的笔记本电脑

Computer2  c = new Laptop2(new Lenovo());

c.sale();

//销售神舟的台式机

Computer2 c2 = new Desktop2(new Shenzhou());

c2.sale();

}

}

# 第239集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季239集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 组合模式（composite）:

## 应用场景：

        -把部分和整体的关系用到树形结构来表示，从而使客户端可以使用统一的方式处理部分对象和整体对象。

## 组合模式核心：

        1）抽象构件（component）角色：定义了叶子和容器构件的共同点。

        2）叶子（leaf）构件角色：无子节点

        3）容器（composite）构件角色：有容器特征，可以包含子节点。

package com.bjsxt.composite;

/\*\*

 \* 抽象组件

 \* @author Administrator

 \*

 \*/

public interface Component {

void operation();

}

//叶子组件

interface Leaf extends Component {

}

//容器组件

interface Composite extends Component {

void add(Component c);

void remove(Component c);

Component getChild(int index);

}

# 使用组合模式，模拟杀毒软件的架构设计！

package com.bjsxt.composite;

import java.util.ArrayList;

import java.util.List;

//抽象构建

public interface AbstractFile {

void killVirus();  //杀毒

}

class ImageFile implements AbstractFile {

private String name;

public ImageFile(String name) {

super();

this.name = name;

}

@Override

public void killVirus() {

System.out.println("---图像文件："+name+",进行查杀！");

}

}

class TextFile implements AbstractFile {

private String name;

public TextFile(String name) {

super();

this.name = name;

}

@Override

public void killVirus() {

System.out.println("---文本文件："+name+",进行查杀！");

}

}

class VideoFile implements AbstractFile {

private String name;

public VideoFile(String name) {

super();

this.name = name;

}

@Override

public void killVirus() {

System.out.println("---视频文件："+name+",进行查杀！");

}

}

class Folder implements AbstractFile {

private String name;

//定义容器，用来存放本容器构建下的子节点

private List<AbstractFile> list = new ArrayList<AbstractFile>();

public Folder(String name) {

super();

this.name = name;

}

public void add(AbstractFile file){

list.add(file);

}

public void remove(AbstractFile file){

list.remove(file);

}

public AbstractFile getChild(int index){

return list.get(index);

}

@Override

public void killVirus() {

System.out.println("---文件夹："+name+",进行查杀");

for (AbstractFile file : list) {

file.killVirus();

}

}

}

## 测试

package com.bjsxt.composite;

public class Client {

public static void main(String[] args) {

AbstractFile f2,f3,f4,f5;

Folder f1 = new Folder("我的收藏");

f2 = new ImageFile("老高的大头像.jpg");

f3 = new TextFile("Hello.txt");

f1.add(f2);

f1.add(f3);

Folder f11 = new Folder("电影");

f4 = new VideoFile("笑傲江湖.avi");

f5 = new VideoFile("神雕侠侣.avi");

f11.add(f4);

f11.add(f5);

f1.add(f11);

// f2.killVirus();

f1.killVirus();

}

}

# 第240集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季240集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

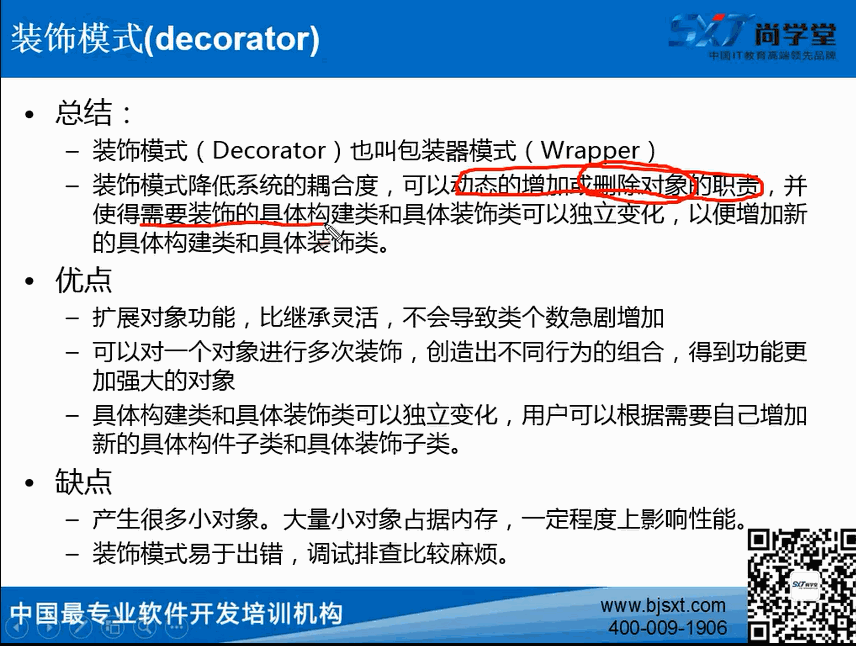
【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 装饰模式（decorator）：

    职责：

    -动态的为一个对象增加新的功能。

    -装饰模式是一个用于代替继承的技术，无须通过继承增加子类就能扩展对象的新功能，使用对象的关联关系代替继承关系，更加灵活，同时避免类型体系的快速膨胀。

   装饰模式的总结：    

## 代码的展示如下：

package com.bjsxt.decorator;

/\*\*

 \* 抽象构建

 \* @author 尚学堂高淇，www.sxt.cn

 \*

 \*/

public interface ICar {

void move();

}

//ConcreteComponent 具体构件角色(真实对象)

class Car implements ICar {

@Override

public void move() {

System.out.println("陆地上跑！");

}

}

//Decorator装饰角色

class SuperCar implements ICar {

protected ICar car;

public SuperCar(ICar car) {

super();

this.car = car;

}

@Override

public void move() {

car.move();

}

}

//ConcreteDecorator具体装饰角色

class FlyCar extends SuperCar {

public FlyCar(ICar car) {

super(car);

}

public void fly(){

System.out.println("天上飞！");

}

@Override

public void move() {

super.move();

fly();

}

}

//ConcreteDecorator具体装饰角色

class WaterCar extends SuperCar {

public WaterCar(ICar car) {

super(car);

}

public void swim(){

System.out.println("水上游！");

}

@Override

public void move() {

super.move();

swim();

}

}

//ConcreteDecorator具体装饰角色

class AICar extends SuperCar {

public AICar(ICar car) {

super(car);

}

public void autoMove(){

System.out.println("自动跑！");

}

@Override

public void move() {

super.move();

autoMove();

}

}

## 测试代码如下：

package com.bjsxt.decorator;

import java.io.FileInputStream;

import java.io.InputStreamReader;

public class Client {

public static void main(String[] args) {

Car car  = new Car();

car.move();

System.out.println("增加新的功能，飞行----------");

FlyCar flycar = new FlyCar(car);

flycar.move();

System.out.println("增加新的功能，水里游---------");

WaterCar  waterCar = new WaterCar(car);

waterCar.move();

System.out.println("增加两个新的功能，飞行，水里游-------");

WaterCar waterCar2 = new WaterCar(new FlyCar(car));

waterCar2.move();

// Reader r = new BufferedReader(new InputStreamReader(new FileInputStream(new File("d:/a.txt"))));

}

}

# 第241集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季241集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 门面模式（外观模式）：

    核心：为了系统提供统一的入口，封装子系统的复杂性，便于客户端调用。

## 代码展示如下：

package com.bjsxt.facade;

public interface 工商局  {

void checkName();  //核名

}

class 海淀区工商局 implements 工商局 {

@Override

public void checkName() {

System.out.println("检查名字是否有冲突！");

}

}

package com.bjsxt.facade;

public interface 税务局 {

void taxCertificate();  //办理税务登记证

}

class 海淀税务局 implements 税务局 {

@Override

public void taxCertificate() {

System.out.println("在海淀税务局办理税务登记证！");

}

}

package com.bjsxt.facade;

public interface 银行 {

void  openAccount();  //开户

}

class 中国工商银行 implements 银行 {

@Override

public void openAccount() {

System.out.println("在中国工商银行开户！");

}

}

package com.bjsxt.facade;

public interface 质检局 {

void  orgCodeCertificate();  //办理组织机构代码证

}

class 海淀质检局 implements 质检局 {

@Override

public void orgCodeCertificate() {

System.out.println("在海淀区质检局办理组织机构代码证！");

}

}

## 门面模式的展示代码如下：

package com.bjsxt.facade;

/\*\*

 \* 办理注册公司流程的门面对象

 \* @author Administrator

 \*

 \*/

public class RegisterFacade {

public void register(){

工商局  a = new 海淀区工商局();

a.checkName();

质检局 b = new 海淀质检局();

b.orgCodeCertificate();

税务局  c  = new 海淀税务局();

c.taxCertificate();

银行  d = new 中国工商银行();

d.openAccount();

}

}

## 测试代码：

package com.bjsxt.facade;

public class Client1 {

public static void main(String[] args) {

// 工商局  a = new 海淀区工商局();

// a.checkName();

// 质检局 b = new 海淀质检局();

// b.orgCodeCertificate();

// 税务局  c  = new 海淀税务局();

// c.taxCertificate();

// 银行  d = new 中国工商银行();

// d.openAccount();

new RegisterFacade().register();

}

}

# 第242集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季242集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 享元模式（FlyWeight）：

## 核心：

        --享元模式一共享的方式高效地支持大量细粒度对象的重用。

        --享元模式对象能 做到共享的关键是区分内部状态和外部状态。

            -内部状态：可以共享，不会随环境变化而变化。

            -外部状态：不可以共享，会随着环境变化而改变。

## 代码展示：

package com.bjsxt.flyweight;

/\*\*

 \* 外部状态UnSharedConcreteFlyWeight

 \* @author Administrator

 \*

 \*/

public class Coordinate {

private int x,y;

public Coordinate(int x, int y) {

super();

this.x = x;

this.y = y;

}

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

}

package com.bjsxt.flyweight;

/\*\*

 \* 享元类

 \* @author Administrator

 \*

 \*/

public interface ChessFlyWeight {

void setColor(String c);

String getColor();

void display(Coordinate c);

}

class ConcreteChess implements ChessFlyWeight {

private String color;

public ConcreteChess(String color) {

super();

this.color = color;

}

@Override

public void display(Coordinate c) {

System.out.println("棋子颜色："+color);

System.out.println("棋子位置："+c.getX()+"----"+c.getY());

}

@Override

public String getColor() {

return color;

}

@Override

public void setColor(String c) {

this.color = c;

}

}

package com.bjsxt.flyweight;

import java.util.HashMap;

import java.util.Map;

/\*\*

 \* 享元工厂类

 \* @author Administrator

 \*

 \*/

public class ChessFlyWeightFactory {

//享元池

private static Map<String,ChessFlyWeight> map = new HashMap<String, ChessFlyWeight>();

public static ChessFlyWeight  getChess(String color){

if(map.get(color)!=null){

return map.get(color);

}else{

ChessFlyWeight cfw = new ConcreteChess(color);

map.put(color, cfw);

return cfw;

}

}

}

## 测试代码：

package com.bjsxt.flyweight;

public class Client {

public static void main(String[] args) {

ChessFlyWeight chess1 = ChessFlyWeightFactory.getChess("黑色");

ChessFlyWeight chess2 = ChessFlyWeightFactory.getChess("黑色");

System.out.println(chess1);

System.out.println(chess2);

System.out.println("增加外部状态的处理===========");

chess1.display(new Coordinate(10, 10));

chess2.display(new Coordinate(20, 20));

}

}

# 第243集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季243集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 责任链模式（chain of responsibility）：

    定义：将能够处理同一类请求的对象连成一条链，所提交的请求将沿着链传递，链上的对象逐个判断是否有能力处理该请求，如果能则处理，如果不能则传递给链上的下一个对象。

    场景：

        -打牌时，轮流出牌

        -接力赛跑

        -大学中，奖学金的审批

        -公司中，公文的审批。

## 代码展示如下：

package com.bjsxt.chainOfResp;

/\*\*

 \* 封装请假的基本信息

 \* @author Administrator

 \*

 \*/

public class LeaveRequest {

private String empName;

private int leaveDays;

private String reason;

public LeaveRequest(String empName, int leaveDays, String reason) {

super();

this.empName = empName;

this.leaveDays = leaveDays;

this.reason = reason;

}

public String getEmpName() {

return empName;

}

public void setEmpName(String empName) {

this.empName = empName;

}

public int getLeaveDays() {

return leaveDays;

}

public void setLeaveDays(int leaveDays) {

this.leaveDays = leaveDays;

}

public String getReason() {

return reason;

}

public void setReason(String reason) {

this.reason = reason;

}

}

package com.bjsxt.chainOfResp;

/\*\*

 \* 抽象类

 \* @author Administrator

 \*

 \*/

public abstract class Leader {

protected String name;

protected Leader nextLeader; //责任链上的后继对象

public Leader(String name) {

super();

this.name = name;

}

//设定责任链上的后继对象

public void setNextLeader(Leader nextLeader) {

this.nextLeader = nextLeader;

}

/\*\*

 \* 处理请求的核心的业务方法

 \* @param request

 \*/

public abstract void handleRequest(LeaveRequest request);

}

package com.bjsxt.chainOfResp;

/\*\*

 \* 经理

 \* @author Administrator

 \*

 \*/

public class Manager extends Leader {

public Manager(String name) {

super(name);

}

@Override

public void handleRequest(LeaveRequest request) {

if(request.getLeaveDays()<10){

System.out.println("员工："+request.getEmpName()+"请假，天数："+request.getLeaveDays()+",理由："+request.getReason());

System.out.println("经理："+this.name+",审批通过！");

}else{

if(this.nextLeader!=null){

this.nextLeader.handleRequest(request);

}

}

}

}

package com.bjsxt.chainOfResp;

/\*\*

 \* 总经理

 \* @author Administrator

 \*/

public class GeneralManager extends Leader {

public GeneralManager(String name) {

super(name);

}

@Override

public void handleRequest(LeaveRequest request) {

if(request.getLeaveDays()<30){

System.out.println("员工："+request.getEmpName()+"请假，天数："+request.getLeaveDays()+",理由："+request.getReason());

System.out.println("总经理："+this.name+",审批通过！");

}else{

System.out.println("莫非"+request.getEmpName()+"想辞职，居然请假"+request.getLeaveDays()+"天！");

}

}

}

package com.bjsxt.chainOfResp;

/\*\*

 \* 主任

 \* @author Administrator

 \*

 \*/

public class Director extends Leader {

public Director(String name) {

super(name);

}

@Override

public void handleRequest(LeaveRequest request) {

if(request.getLeaveDays()<3){

System.out.println("员工："+request.getEmpName()+"请假，天数："+request.getLeaveDays()+",理由："+request.getReason());

System.out.println("主任："+this.name+",审批通过！");

}else{

if(this.nextLeader!=null){

this.nextLeader.handleRequest(request);

}

}

}

}

package com.bjsxt.chainOfResp;

/\*\*

 \* 副总经理

 \* @author Administrator

 \*

 \*/

public class ViceGeneralManager extends Leader {

public ViceGeneralManager(String name) {

super(name);

}

@Override

public void handleRequest(LeaveRequest request) {

if(request.getLeaveDays()<20){

System.out.println("员工："+request.getEmpName()+"请假，天数："+request.getLeaveDays()+",理由："+request.getReason());

System.out.println("副总经理："+this.name+",审批通过！");

}else{

if(this.nextLeader!=null){

this.nextLeader.handleRequest(request);

}

}

}

}

## 测试代码如下：

package com.bjsxt.chainOfResp;

public class Client {

public static void main(String[] args) {

Leader a = new Director("张三");

Leader b = new Manager("李四");

Leader b2 = new ViceGeneralManager("李小四");

Leader c = new GeneralManager("王五");

//组织责任链对象的关系

a.setNextLeader(b);

b.setNextLeader(b2);

b2.setNextLeader(c);

//开始请假操作

LeaveRequest req1 = new LeaveRequest("TOM", 15, "回英国老家探亲！");

a.handleRequest(req1);

}

}

# 第244集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季244集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 迭代器模式（Iterator）：

    迭代器模式我们不陌生，我们jdk提供的也有这样啊案例，如下是我们自定义的迭代器模式的代码如下：

package com.bjsxt.iterator;

import java.util.ArrayList;

import java.util.List;

/\*\*

 \* 自定义的聚合类

 \* @author Administrator

 \*

 \*/

public class ConcreteMyAggregate {

private List<Object> list = new ArrayList<Object>();

public void addObject(Object obj){

this.list.add(obj);

}

public void removeObject(Object obj){

this.list.remove(obj);

}

public List<Object> getList() {

return list;

}

public void setList(List<Object> list) {

this.list = list;

}

//获得迭代器

public MyIterator  createIterator(){

return new ConcreteIterator();

}

//使用内部类定义迭代器，可以直接使用外部类的属性

private class ConcreteIterator implements MyIterator {

private int cursor;  //定义游标用于记录遍历时的位置

@Override

public void first() {

cursor = 0;

}

@Override

public Object getCurrentObj() {

return list.get(cursor);

}

@Override

public boolean hasNext() {

if(cursor<list.size()){

return true;

}

return false;

}

@Override

public boolean isFirst() {

return cursor==0?true:false;

}

@Override

public boolean isLast() {

return cursor==(list.size()-1)?true:false;

}

@Override

public void next() {

if(cursor<list.size()){

cursor++;

}

}

}

}

## 迭代器接口：

package com.bjsxt.iterator;

/\*\*

 \*  自定义的迭代器接口

 \* @author Administrator

 \*

 \*/

public interface MyIterator {

void first(); //将游标指向第一个元素

void next(); //将游标指向下一个元素

boolean hasNext();//判断是否存在下一个元素

boolean isFirst();

boolean isLast();

Object getCurrentObj();  //获取当前游标指向的对象

}

## 测试代码：

package com.bjsxt.iterator;

public class Client {

public static void main(String[] args) {

ConcreteMyAggregate cma = new ConcreteMyAggregate();

cma.addObject("aa");

cma.addObject("bb");

cma.addObject("cc");

MyIterator iter = cma.createIterator();

while(iter.hasNext()){

System.out.println(iter.getCurrentObj());

iter.next();

}

}

}

# 第245集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季245集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 中介者模式（Mediator）：

package com.bjsxt.mediator;

public interface Mediator {

void register(String dname,Department d);

void command(String dname);

}

package com.bjsxt.mediator;

public class Market implements Department {

private Mediator m;  //持有中介者(总经理)的引用

public Market(Mediator m) {

super();

this.m = m;

m.register("market", this);

}

@Override

public void outAction() {

System.out.println("汇报工作！项目承接的进度，需要资金支持！");

m.command("finacial");

}

@Override

public void selfAction() {

System.out.println("跑去接项目！");

}

}

package com.bjsxt.mediator;

public class Finacial implements Department {

private Mediator m;  //持有中介者(总经理)的引用

public Finacial(Mediator m) {

super();

this.m = m;

m.register("finacial", this);

}

@Override

public void outAction() {

System.out.println("汇报工作！没钱了，钱太多了！怎么花?");

}

@Override

public void selfAction() {

System.out.println("数钱！");

}

}

package com.bjsxt.mediator;

public class Development implements Department {

private Mediator m;  //持有中介者(总经理)的引用

public Development(Mediator m) {

super();

this.m = m;

m.register("development", this);

}

@Override

public void outAction() {

System.out.println("汇报工作！没钱了，需要资金支持！");

}

@Override

public void selfAction() {

System.out.println("专心科研，开发项目！");

}

}

package com.bjsxt.mediator;

//同事类的接口

public interface Department {

void selfAction(); //做本部门的事情

void outAction();  //向总经理发出申请

}

package com.bjsxt.mediator;

import java.util.HashMap;

import java.util.Map;

public class President implements Mediator {

private Map<String,Department> map = new HashMap<String , Department>();

@Override

public void command(String dname) {

map.get(dname).selfAction();

}

@Override

public void register(String dname, Department d) {

map.put(dname, d);

}

}

package com.bjsxt.mediator;

public class Client {

public static void main(String[] args) {

Mediator m = new President();

Market   market = new Market(m);

Development devp = new Development(m);

Finacial f = new Finacial(m);

market.selfAction();

market.outAction();

}

}

# 第246集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季246集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 命令模式（command）:

    将一个请求封装为一个对象，从而使我们可用不同的请求对客户进行参数化；对请求排队或者记录请求日志，以及支持可以撤销的操作，也称之为：动作Action模式，事务transaction模式。

# 结构：

   - Command抽象命令类

    -ConcreterCommand具体命令类

    -Invoker调用者/请求者

    -Receiver接收者

    -Client客户类

## 代码展示如下：

package com.bjsxt.command;

public interface Command {

/\*\*

 \* 这个方法是一个返回结果为空的方法。

 \* 实际项目中，可以根据需求设计多个不同的方法

 \*/

void execute();

}

class ConcreteCommand implements Command {

private Receiver receiver; //命令的真正的执行者

public ConcreteCommand(Receiver receiver) {

super();

this.receiver = receiver;

}

@Override

public void execute() {

//命令真正执行前或后，执行相关的处理！

receiver.action();

}

}

package com.bjsxt.command;

//调用者/发起者

public class Invoke {

private Command command;   //也可以通过容器List<Command>容纳很多命令对象，进行批处理。数据库底层的事务管理就是类似的结构！

public Invoke(Command command) {

super();

this.command = command;

}

//业务方法 ，用于调用命令类的方法

public void call(){

command.execute();

}

}

package com.bjsxt.command;

/\*\*

 \* 真正的命令的执行者

 \* @author Administrator

 \*

 \*/

public class Receiver {

public void action(){

System.out.println("Receiver.action()");

}

}

package com.bjsxt.command;

public class Client {

public static void main(String[] args) {

Command c = new ConcreteCommand(new Receiver());

Invoke i = new Invoke(c);

i.call();

// new Receiver().action();

}

}

# 第247集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季247集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 解释器模式（Interpreter）：  简单了解一下即可！

    -是一种不常用的设计模式

    -用于描述如何构成一个简单的语言解释器，主要用于使用面向对象语言开发的编译器和解释器设计。

    -当我们需要开发一种新的语言时，可以考虑使用解释器模式。

# 开发中常见的场景:

    -EL表达式式的处理

    -正则表达式解释器。

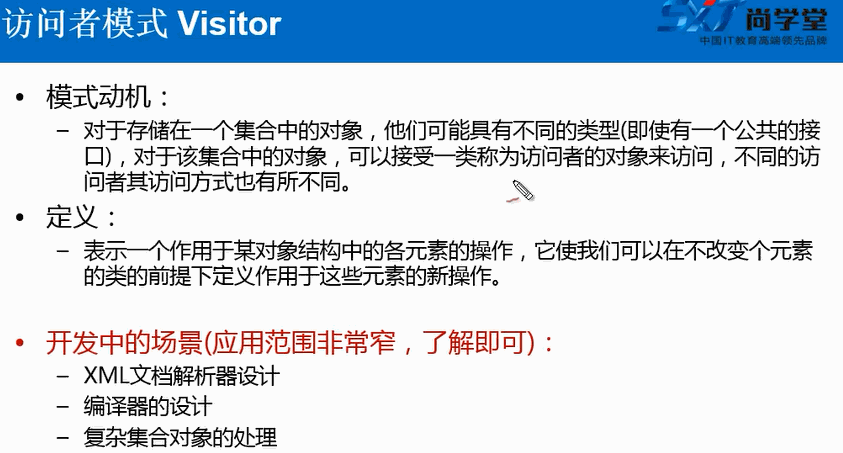
    -SQL语法的计数器。

    -数学表达式解析器

        。如现成的工具包：Math Expression String Parser ,Expression4J等。

 访问者模式（Visitor）:一般了解

  .模式动机：



# 第248集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季248集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 策略模式（strategy）：

package com.bjsxt.strategy;

/\*\*

 \* 实现起来比较容易，符合一般开发人员的思路

 \* 假如，类型特别多，算法比较复杂时，整个条件语句的代码就变得很长，难于维护。

 \* 如果有新增类型，就需要频繁的修改此处的代码！

 \* 不符合开闭原则！

 \* @author Administrator

 \*

 \*/

public class TestStrategy {

public double getPrice(String type, double price) {

if (type.equals("普通客户小批量")) {

System.out.println("不打折,原价");

return price;

} else if (type.equals("普通客户大批量")) {

System.out.println("打九折");

return price \* 0.9;

} else if (type.equals("老客户小批量")) {

System.out.println("打八五折");

return price \* 0.85;

} else if (type.equals("老客户大批量")) {

System.out.println("打八折");

return price \* 0.8;

}

return price;

}

}

package com.bjsxt.strategy;

public interface Strategy {

public double getPrice(double  standardPrice);

}

## 接口的实现类如下：

package com.bjsxt.strategy;

public class NewCustomerFewStrategy implements Strategy {

@Override

public double getPrice(double standardPrice) {

System.out.println("不打折，原价");

return standardPrice;

}

}

package com.bjsxt.strategy;

public class NewCustomerManyStrategy implements Strategy {

@Override

public double getPrice(double standardPrice) {

System.out.println("打九折");

return standardPrice\*0.9;

}

}

package com.bjsxt.strategy;

public class OldCustomerFewStrategy implements Strategy {

@Override

public double getPrice(double standardPrice) {

System.out.println("打八五折");

return standardPrice\*0.85;

}

}

package com.bjsxt.strategy;

public class OldCustomerManyStrategy implements Strategy {

@Override

public double getPrice(double standardPrice) {

System.out.println("打八折");

return standardPrice\*0.8;

}

}

package com.bjsxt.strategy;

/\*\*

 \* 负责和具体的策略类交互

 \* 这样的话，具体的算法和直接的客户端调用分离了，使得算法可以独立于客户端独立的变化。

 \* 如果使用spring的依赖注入功能，还可以通过配置文件，动态的注入不同策略对象，动态的切换不同的算法.

 \* @author Administrator

 \*

 \*/

public class Context {

private Strategy strategy; //当前采用的算法对象

//可以通过构造器来注入

public Context(Strategy strategy) {

super();

this.strategy = strategy;

}

//可以通过set方法来注入

public void setStrategy(Strategy strategy) {

this.strategy = strategy;

}

public void pringPrice(double s){

System.out.println("您该报价："+strategy.getPrice(s));

}

}

package com.bjsxt.strategy;

public class Client {

public static void main(String[] args) {

Strategy s1 = new OldCustomerManyStrategy();

Context ctx = new Context(s1);

ctx.pringPrice(998);

}

}

# 第249集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季249集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 模板方法模式（template method）:

package com.bjsxt.templateMethod;

public abstract class BankTemplateMethod {

//具体方法

public void takeNumber(){

System.out.println("取号排队");

}

public abstract void transact(); //办理具体的业务 //钩子方法

public void evaluate(){

System.out.println("反馈评分");

}

public final void process(){ //模板方法！！！

this.takeNumber();

this.transact();

this.evaluate();

}

}

package com.bjsxt.templateMethod;

public class Client {

public static void main(String[] args) {

BankTemplateMethod btm = new DrawMoney();

btm.process();

//采用匿名内部类

BankTemplateMethod btm2 = new BankTemplateMethod() {

@Override

public void transact() {

System.out.println("我要存钱！");

}

};

btm2.process();

BankTemplateMethod btm3 = new BankTemplateMethod() {

@Override

public void transact() {

System.out.println("我要理财！我这里有2000万韩币");

}

};

btm3.process();

}

}

class DrawMoney extends BankTemplateMethod {

@Override

public void transact() {

System.out.println("我要取款！！！");

}

}

# 第250集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季250集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 状态模式（State）：

    核心：用于解决系统中复杂对象状态转换以及不同状态下行为的封装问题。

    结构：

    -Context环境类

            .环境类中维护一个State对象，他是定义来了当前的状态。

    -State抽象状态类

    -ConcreteState具体状态类

        .每一个类封装了一个状态对应的行为。

package com.bjsxt.state;

public interface State {

void handle();

}

## 接口的实现类如下：

package com.bjsxt.state;

/\*\*

 \* 空闲状态

 \*/

public class FreeState implements State {

@Override

public void handle() {

System.out.println("房间空闲！！！没人住！");

}

}

package com.bjsxt.state;

/\*\*

 \* 已入住状态

 \* @author Administrator

 \*

 \*/

public class CheckedInState implements State {

@Override

public void handle() {

System.out.println("房间已入住！请勿打扰！");

}

}

package com.bjsxt.state;

/\*\*

 \* 已预订状态

 \*/

public class BookedState implements State {

@Override

public void handle() {

System.out.println("房间已预订！别人不能定！");

}

}

package com.bjsxt.state;

/\*\*

 \* 已预订状态

 \*/

public class BookedState implements State {

@Override

public void handle() {

System.out.println("房间已预订！别人不能定！");

}

}

package com.bjsxt.state;

/\*\*

 \* 房间对象

 \* @author Administrator

 \*

 \*/

public class HomeContext {

//如果是银行系统，这个Context类就是账号。根据金额不同，切换不同的状态！

private State state;

public void setState(State s){

System.out.println("修改状态！");

state = s;

state.handle();

}

}

## 测试类：

package com.bjsxt.state;

public class Client {

public static void main(String[] args) {

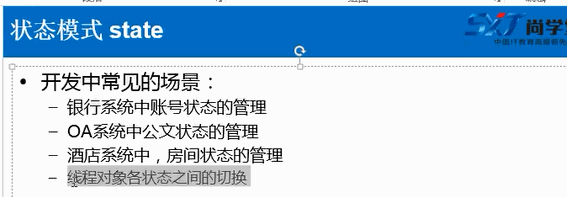
HomeContext ctx = new HomeContext();

ctx.setState(new FreeState());

ctx.setState(new BookedState());

}

}



# 第251集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季251集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 观察者模式（Observer）；

package com.bjsxt.observer;

public interface Observer {

void  update(Subject subject);

}

package com.bjsxt.observer;

import java.util.ArrayList;

import java.util.List;

public class Subject {

protected List<Observer> list = new ArrayList<Observer>();

public void registerObserver(Observer obs){

list.add(obs);

}

public void removeObserver(Observer obs){

list.add(obs);

}

//通知所有的观察者更新状态

public void notifyAllObservers(){

for (Observer obs : list) {

obs.update(this);

}

}

}

package com.bjsxt.observer;

public class ConcreteSubject extends Subject {

private int state;

public int getState() {

return state;

}

public void setState(int state) {

this.state = state;

//主题对象(目标对象)值发生了变化，请通知所有的观察者

this.notifyAllObservers();

}

}

## 测试类代码如下：

package com.bjsxt.observer;

public class Client {

public static void main(String[] args) {

//目标对象

ConcreteSubject subject = new ConcreteSubject();

//创建多个观察者

ObserverA  obs1 = new ObserverA();

ObserverA  obs2 = new ObserverA();

ObserverA  obs3 = new ObserverA();

//将这三个观察者添加到subject对象的观察者队伍中

subject.registerObserver(obs1);

subject.registerObserver(obs2);

subject.registerObserver(obs3);

//改变subject的状态

subject.setState(3000);

System.out.println("########################");

//我们看看，观察者的状态是不是也发生了变化

System.out.println(obs1.getMyState());

System.out.println(obs2.getMyState());

System.out.println(obs3.getMyState());

//改变subject的状态

subject.setState(30);

System.out.println("########################");

//我们看看，观察者的状态是不是也发生了变化

System.out.println(obs1.getMyState());

System.out.println(obs2.getMyState());

System.out.println(obs3.getMyState());

}

}

# 第252集

尚学堂JAVA最全教程

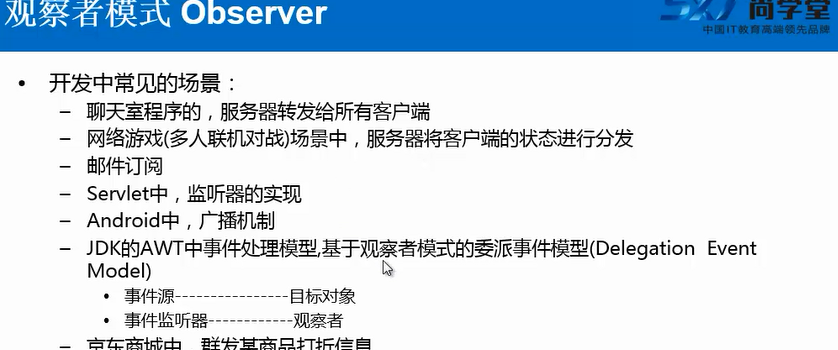
配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季252集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 观察者模式进阶（jdk）:



package com.bjsxt.observer2;

import java.util.Observable;

//目标对象

public class ConcreteSubject extends Observable {

private int state;

public void set(int s){

state = s;  //目标对象的状态发生了改变

setChanged();  //表示目标对象已经做了更改

notifyObservers(state);  //通知所有的观察者

}

public int getState() {

return state;

}

public void setState(int state) {

this.state = state;

}

}

package com.bjsxt.observer2;

import java.util.Observable;

import java.util.Observer;

public class ObserverA implements Observer {

private int myState;

@Override

public void update(Observable o, Object arg) {

myState = ((ConcreteSubject)o).getState();

}

public int getMyState() {

return myState;

}

public void setMyState(int myState) {

this.myState = myState;

}

}

## 测试类

package com.bjsxt.observer2;

public class Client {

public static void main(String[] args) {

//创建目标对象Obserable

ConcreteSubject subject = new ConcreteSubject();

//创建观察者

ObserverA obs1 = new ObserverA();

ObserverA obs2 = new ObserverA();

ObserverA obs3 = new ObserverA();

//将上面三个观察者对象添加到目标对象subject的观察者容器中

subject.addObserver(obs1);

subject.addObserver(obs2);

subject.addObserver(obs3);

//改变subject对象的状态

subject.set(3000);

System.out.println("===============状态修改了！");

//观察者的状态发生了变化

System.out.println(obs1.getMyState());

System.out.println(obs2.getMyState());

System.out.println(obs3.getMyState());

subject.set(600);

System.out.println("===============状态修改了！");

//观察者的状态发生了变化

System.out.println(obs1.getMyState());

System.out.println(obs2.getMyState());

System.out.println(obs3.getMyState());

}

}

# 第253集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季253集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 备忘录模式（memento）；

## 核心：

        -就是保存某个对象内部状态的拷贝，这样以后就可以将该对象恢复到原先的状态。

## 结构：

       -源发器类Originator

       -备忘录类Memento

       -负责人类CareTake

## 备忘点较多时：

        -将备忘录压栈

        -将多个备忘录对象，序列化和持久化。

package com.bjsxt.memento;

/\*\*

 \* 备忘录类

 \* @author Administrator

 \*

 \*/

public class EmpMemento {

private String ename;

private int age;

private double salary;

public EmpMemento(Emp e) {

this.ename = e.getEname();

this.age = e.getAge();

this.salary = e.getSalary();

}

public String getEname() {

return ename;

}

public void setEname(String ename) {

this.ename = ename;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

package com.bjsxt.memento;

/\*\*

 \* 源发器类

 \* @author Administrator

 \*

 \*/

public class Emp {

private String ename;

private int age;

private double salary;

//进行备忘操作，并返回备忘录对象

public EmpMemento  memento(){

return new EmpMemento(this);

}

//进行数据恢复，恢复成制定备忘录对象的值

public void recovery(EmpMemento mmt){

this.ename = mmt.getEname();

this.age = mmt.getAge();

this.salary = mmt.getSalary();

}

public Emp(String ename, int age, double salary) {

super();

this.ename = ename;

this.age = age;

this.salary = salary;

}

public String getEname() {

return ename;

}

public void setEname(String ename) {

this.ename = ename;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

package com.bjsxt.memento;

/\*\*

 \* 负责人类

 \* 负责管理备忘录对象

 \* @author Administrator

 \*

 \*/

public class CareTaker {

private EmpMemento memento;

// private List<EmpMemento> list = new ArrayList<EmpMemento>();

public EmpMemento getMemento() {

return memento;

}

public void setMemento(EmpMemento memento) {

this.memento = memento;

}

}

## 测试类：

package com.bjsxt.memento;

public class Client {

public static void main(String[] args) {

CareTaker taker = new CareTaker();

Emp emp = new Emp("高淇", 18, 900);

System.out.println("第一次打印对象："+emp.getEname()+"---"+emp.getAge()+"---"+emp.getSalary());

taker.setMemento(emp.memento());   //备忘一次

emp.setAge(38);

emp.setEname("搞起");

emp.setSalary(9000);

System.out.println("第二次打印对象："+emp.getEname()+"---"+emp.getAge()+"---"+emp.getSalary());

emp.recovery(taker.getMemento()); //恢复到备忘录对象保存的状态

System.out.println("第三次打印对象："+emp.getEname()+"---"+emp.getAge()+"---"+emp.getSalary());

}

}

# 第254集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季254集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 正则表达式基本知识：

## 基本语法

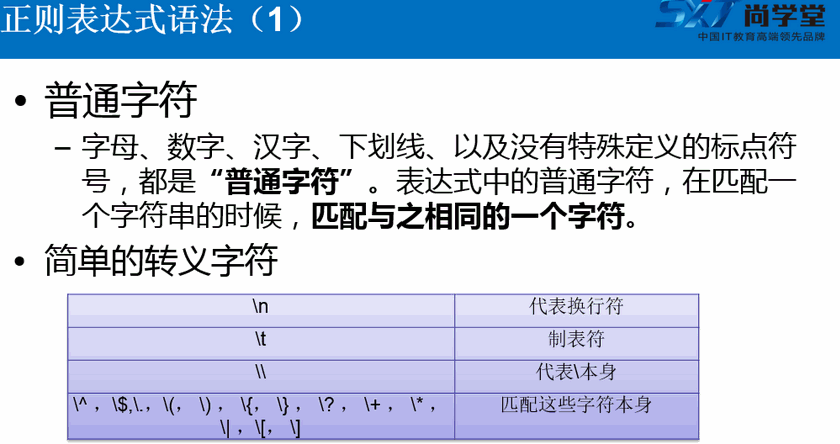
## 高级语法

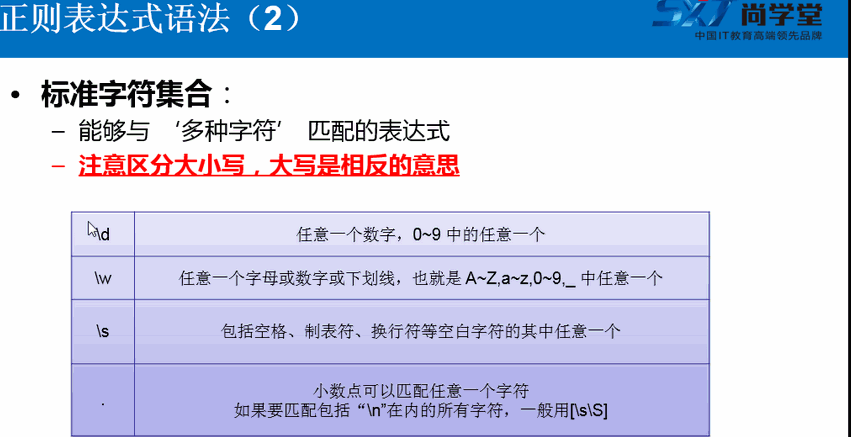
    editplus,notpad++,ultraedit,

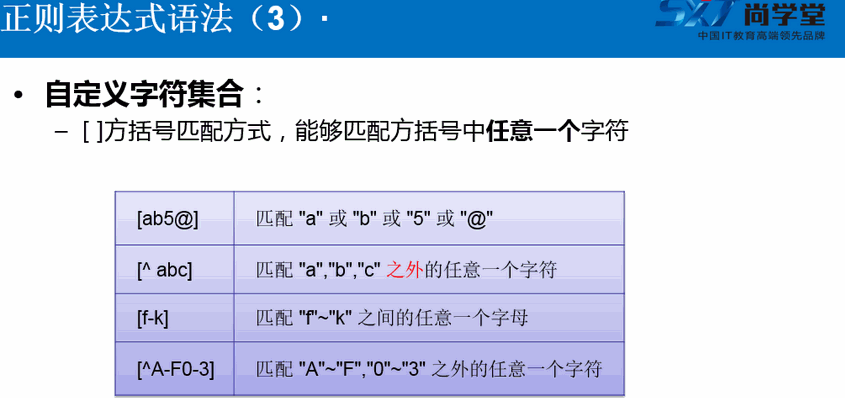
    操作复杂的文本：

    (1),普通字符：

        字母，数字，汉字，下划线，以及没有特殊定义的标点符号，都是“普通字符”。表达式中的普通字符，在匹配一个字符串的时候，匹配与之相同的一个字符。



    (2),基本语法2 

    (3)基本语法3        

# 第255集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

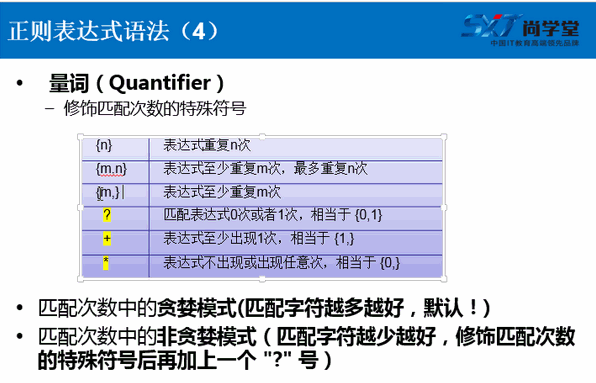
本文档配合：java300集第一季255集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

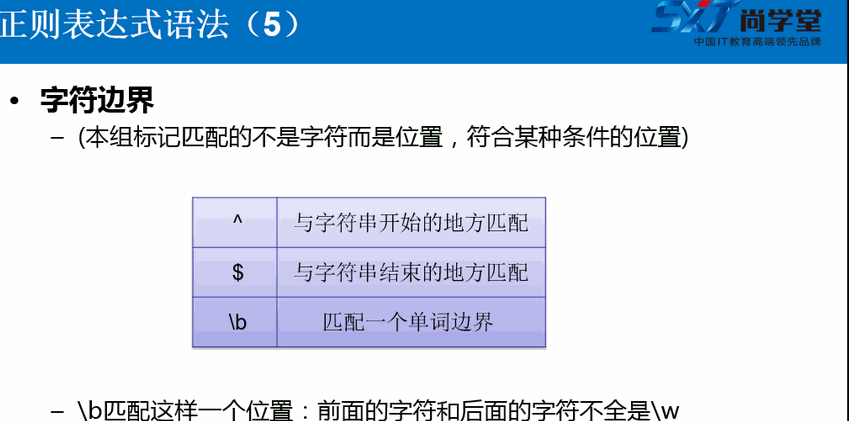
【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 正则表达式

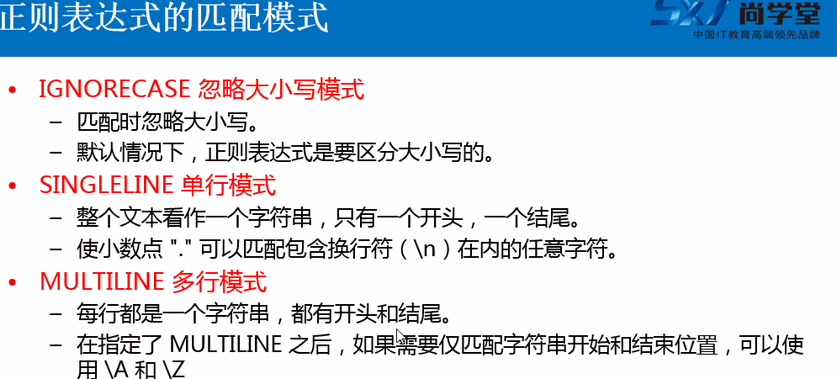
    (1)基本语法4



    (2)基本语法5



    (3)基本语法6



# 第257集

尚学堂JAVA最全教程

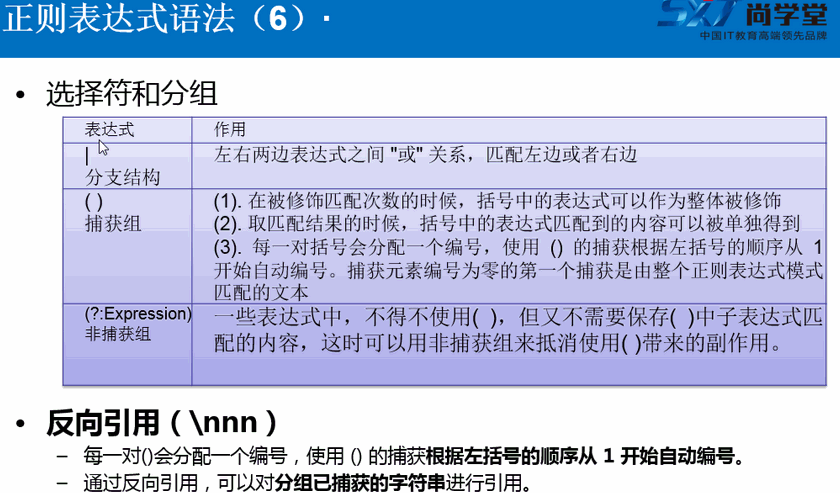
配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季257集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 正则语法之选择符和分组以及反向引用。



# 第258集

尚学堂JAVA最全教程

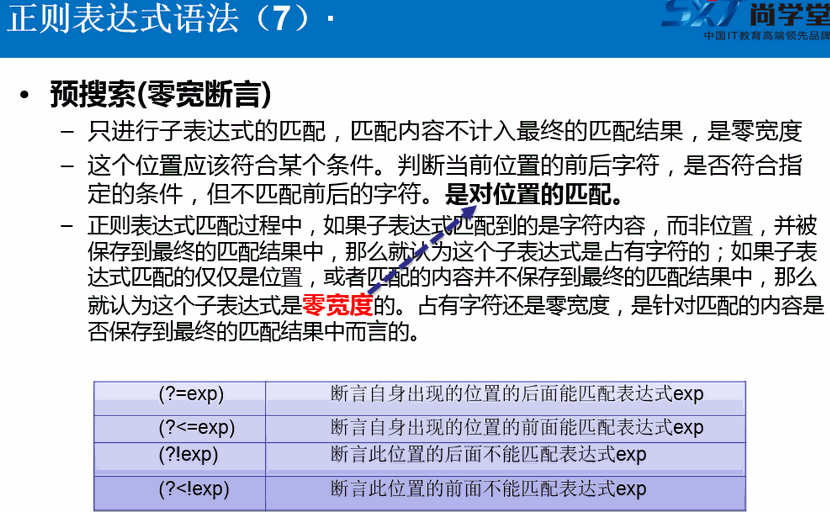
配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季258集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 正则语法7:



# 第259集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季259集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

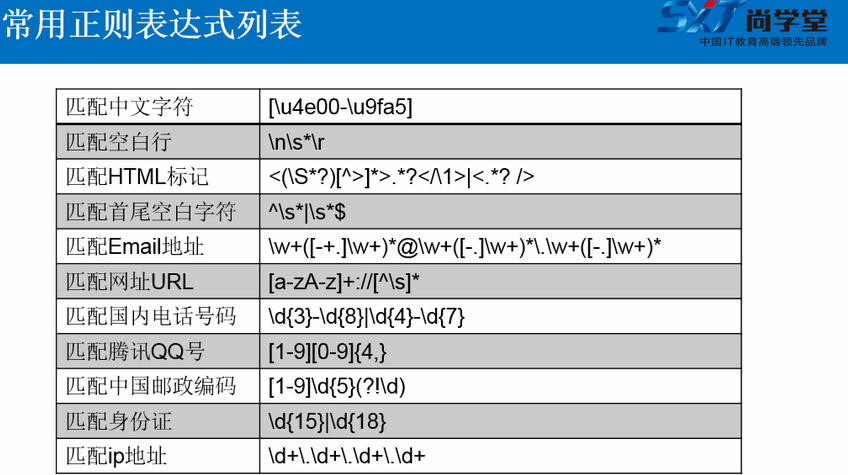
【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 正则的练习：http://sxt.cn/editor/attached/image/20151214/20151214153838_333.png

答案如：

0\d{2,3}-\d{7,9}|1[358]\d{9}

# 常见的正则的表达式：



# 第260集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

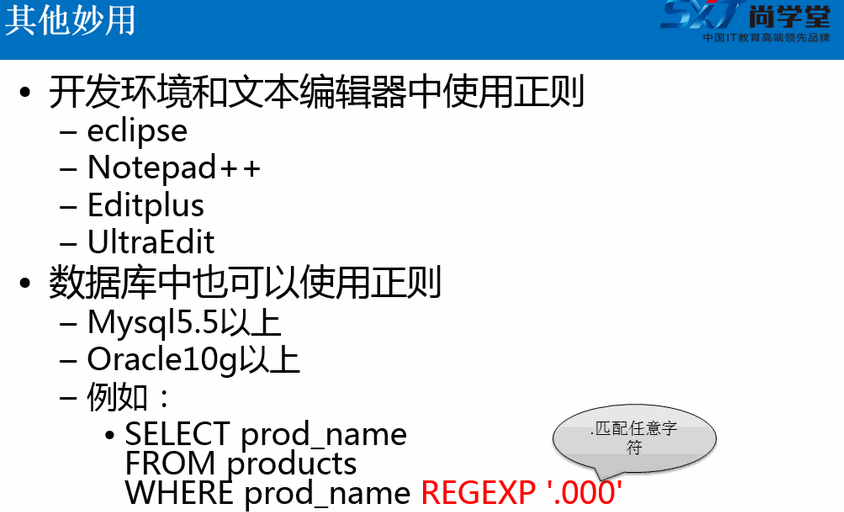
本文档配合：java300集第一季260集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 正则的使用：

    正则的应用没有代码的展示，观看视频练习一下即可：



# 第261集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季261集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Java程序中如何使用正则表达式：

package com.bjsxt.regex.test;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

 \* 测试正则表达式对象的基本用法

 \* @author Administrator

 \*

 \*/

public class Demo01 {

public static void main(String[] args) {

//在这个字符串：asfsdf23323，是否符合指定的正则表达式：\w+

//表达式对象

Pattern p = Pattern.compile("\\w+");

//创建Matcher对象

Matcher m = p.matcher("asfsdf2&&3323");

// boolean yesorno = m.matches(); //尝试将整个字符序列与该模式匹配

// System.out.println(yesorno);

// boolean yesorno2 = m.find(); //该方法扫描输入的序列，查找与该模式匹配的下一个子序列

// System.out.println(yesorno2);

// System.out.println(m.find());

// System.out.println(m.group());

// System.out.println(m.find());

// System.out.println(m.group());

while(m.find()){

System.out.println(m.group()); //group(),group(0)匹配整个表达式的子字符串

System.out.println(m.group(0));

}

}

}

# 正则对象中分组的处理：

package com.bjsxt.regex.test;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

 \* 测试正则表达式对象中分组的处理

 \* @author Administrator

 \*

 \*/

public class Demo02 {

public static void main(String[] args) {

//在这个字符串：asfsdf23323，是否符合指定的正则表达式：\w+

//表达式对象

Pattern p = Pattern.compile("([a-z]+)([0-9]+)");

//创建Matcher对象

Matcher m = p.matcher("aa232\*\*ssd445\*sds223");

while(m.find()){

System.out.println(m.group()); //group(),group(0)匹配整个表达式的子字符串

System.out.println(m.group(1));

System.out.println(m.group(2));

}

}

}

# 正则对象的替换操作

package com.bjsxt.regex.test;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

 \* 测试正则表达式对象的替换操作

 \* @author Administrator

 \*

 \*/

public class Demo03 {

public static void main(String[] args) {

//表达式对象

Pattern p = Pattern.compile("[0-9]");

//创建Matcher对象

Matcher m = p.matcher("aa232\*\*ssd445\*sds223");

//替换

String newStr = m.replaceAll("#");

System.out.println(newStr);

}

}

# 正则对象对分割字符串的操作：

package com.bjsxt.regex.test;

import java.util.Arrays;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

 \* 测试正则表达式对象的分割字符串的操作

 \* @author Administrator

 \*

 \*/

public class Demo04 {

public static void main(String[] args) {

String str = "a232b4334c3434";

String[] arrs = str.split("\\d+");

System.out.println(Arrays.toString(arrs));

}

}

# 第262集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季262集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 正则对象的应用。

    正则主要是处理一下复杂的文本，网络爬虫的原理的代码如下：

package com.bjsxt.regex.test;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.MalformedURLException;

import java.net.URL;

import java.nio.charset.Charset;

import java.util.ArrayList;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

 \* 网络爬虫取链接

 \* @author Administrator

 \*

 \*/

public class WebSpiderTest {

/\*\*

 \* 获得urlStr对应的网页的源码内容

 \* @param urlStr

 \* @return

 \*/

public static String  getURLContent(String urlStr,String charset){

StringBuilder sb = new StringBuilder();

try {

URL url = new URL(urlStr);

BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream(),Charset.forName(charset)));

String temp = "";

while((temp=reader.readLine())!=null){

sb.append(temp);

}

} catch (MalformedURLException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return sb.toString();

}

public static List<String> getMatherSubstrs(String destStr,String regexStr){

Pattern p = Pattern.compile(regexStr); //取到的超链接的地址

Matcher m = p.matcher(destStr);

List<String> result = new ArrayList<String>();

while(m.find()){

result.add(m.group(1));

}

return result;

}

public static void main(String[] args) {

String destStr = getURLContent("http://www.163.com","gbk");

// Pattern p = Pattern.compile("<a[\\s\\S]+?</a>"); //取到的超链接的整个内容

List<String> result = getMatherSubstrs(destStr, "href=\"([\\w\\s./:]+?)\"");

for (String temp : result) {

System.out.println(temp);

}

}

}

# 第263集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季263集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

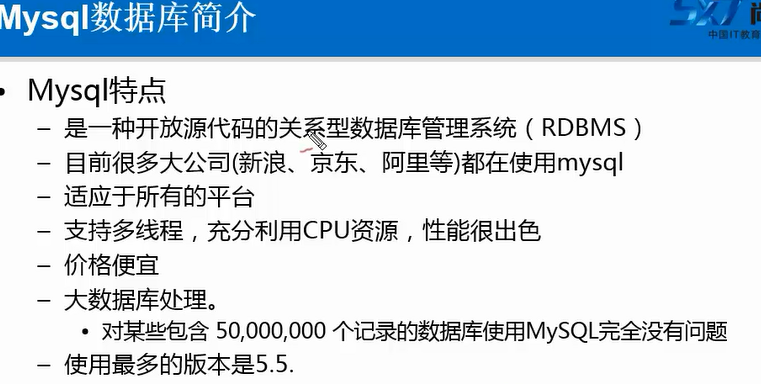
【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC数据库的操作：

     这一集主要是对Mysql数据库的下载，安装，以及安装的注意事项：可以去参考

    本集视频即可了：没有什么代码和思考的东西，多安装几次即可了。





# Mysql的下载：

        -官方主页：<http://www.mysql.com/>

# 第264集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季264集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Mysql的客户端软件Navicat软件的使用：

    Navicat安装

    Navicat的基本使用方式

    -创建数据库

    -建表

    -查询

    -导入sql数据

    -导出查询结果

# Navicat的常见操作，请观看"三百集"视频即可。

# 第265集

尚学堂JAVA最全教程

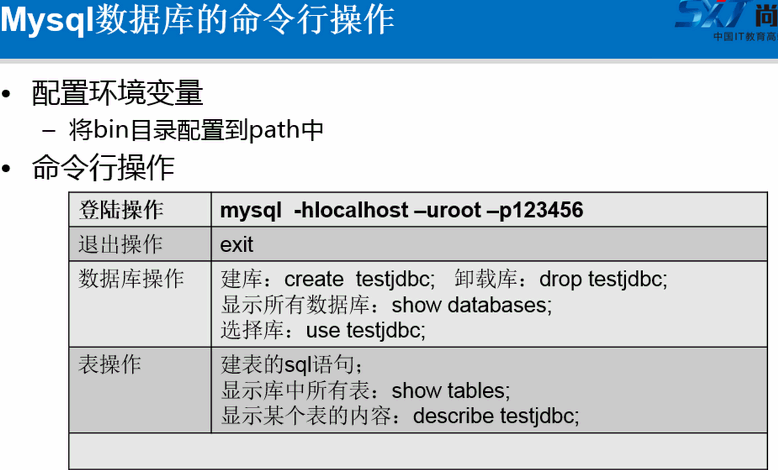
配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季265集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# mysql数据库的命令的行操作：



# 常见的操作如下：

    创建库的命令为：create database test;

    切换到数据库test下命令：use test;

    显示所有数据库命令为：show databases;

    显示库中的表命令为：show tables;

    显示某个表的内容：describe testjdbc;

   更为具体的数据库操作可以参考具体的我们的视频。

# 第266集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

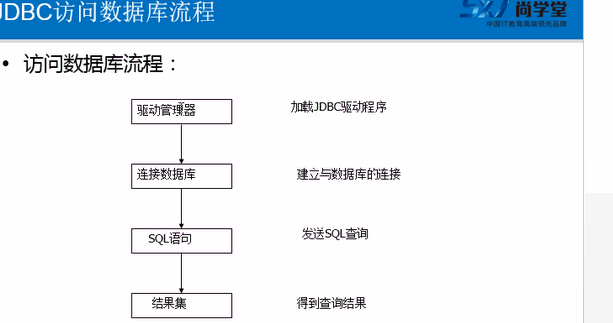
本文档配合：java300集第一季266集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC技术（Java Database Connection）；

JDBC是SUN公司提供的接口，各個数据库去对jdbc进行实现。建立数据库的连接有一下几个步骤：



# 连接数据的代码如下：

package com.bjsxt.jdbc;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

/\*\*

 \* 测试跟数据库建立连接

 \*/

public class Demo01 {

public static void main(String[] args) {

Connection conn = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

long start = System.currentTimeMillis();

//建立连接(连接对象内部其实包含了Socket对象，是一个远程的连接。比较耗时！这是Connection对象管理的一个要点！)

//真正开发中，为了提高效率，都会使用连接池来管理连接对象！

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456s");

long end = System.currentTimeMillis();

System.out.println(conn);

System.out.println("建立连接，耗时："+(end-start)+"ms毫秒");

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 第267集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季267集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC技术

    这一讲主要是Statement语句和sql注入的注意事项：

package com.bjsxt.jdbc;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

 \* 测试Statement接口的用法，执行SQL语句，以及SQL注入问题

 \* @author Administrator

 \*

 \*/

public class Demo02 {

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

//建立连接(连接对象内部其实包含了Socket对象，是一个远程的连接。比较耗时！这是Connection对象管理的一个要点！)

//真正开发中，为了提高效率，都会使用连接池来管理连接对象！

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

stmt = conn.createStatement();

// String name = "赵六";

// String sql = "insert into t\_user (username,pwd,regTime) values ('"+name+"',66666,now())";

// stmt.execute(sql);

//测试SQL注入

String id = "5 or 1=1 ";

String sql = "delete from t\_user where id="+id;

stmt.execute(sql);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

if(stmt!=null){

stmt.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 第268集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季268集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC技术

    PrepredStatement推荐使用的类：

    优点：1）预防sql注入：

           2）内置的处理，效率高！

package com.bjsxt.jdbc;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.text.html.HTMLDocument.HTMLReader.PreAction;

/\*\*

 \* 测试PreparedStatement的基本用法

 \* @author 高淇 www.sxt.cn

 \*

 \*/

public class Demo03 {

public static void main(String[] args) {

Connection conn = null;

PreparedStatement ps = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

String sql = "insert into t\_user (username,pwd,regTime) values (?,?,?)";  //?占位符

ps = conn.prepareStatement(sql);

// ps.setString(1, "高淇3"); //参数索引是从1开始计算， 而不是0

// ps.setString(2, "123456");

// ps.setDate(3, new java.sql.Date(System.currentTimeMillis()));

//可以使用setObject方法处理参数

ps.setObject(1, "高淇5");

ps.setObject(2, "234567");

ps.setObject(3, new java.sql.Date(System.currentTimeMillis()));

System.out.println("插入一行记录");

// ps.execute();

int count = ps.executeUpdate();

System.out.println(count);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 第269集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季269集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC技术

## ResultSet结果集的处理：

package com.bjsxt.jdbc;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

 \* 测试ResultSet结果集的基本用法

 \* @author 高淇 www.sxt.cn

 \*

 \*/

public class Demo04 {

public static void main(String[] args) {

Connection conn = null;

PreparedStatement ps = null;

ResultSet rs = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

String sql = "select id,username,pwd from t\_user where id>?";  //?占位符

ps = conn.prepareStatement(sql);

ps.setObject(1, 2);  //把id大于2的记录都取出来

rs = ps.executeQuery();

while(rs.next()){

System.out.println(rs.getInt(1)+"---"+rs.getString(2)+"---"+rs.getString(3));

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}finally{

//遵循：resultset-->statment-->connection这样的关闭顺序！一定要将三个trycatch块，分开写！

try {

if(rs!=null){

rs.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 第270集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季270集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC批处理技术：

     批处理我们推荐使用Statement,可以大大提高我们的效率！代码展示如下：

       注意事项：

            1）使用Statement,

            2)conn.setAutoCommit(false);

package com.bjsxt.jdbc;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

 \* 测试批处理的基本用法

 \* @author 高淇 www.sxt.cn

 \*/

public class Demo05 {

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

ResultSet rs = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

conn.setAutoCommit(false);  //设为手动提交

long start = System.currentTimeMillis();

stmt = conn.createStatement();

for(int i=0;i<20000;i++){

stmt.addBatch("insert into t\_user (username,pwd,regTime) values ('gao"+i+"',666666,now())");

}

stmt.executeBatch();

conn.commit();  //提交事务

long end = System.currentTimeMillis();

System.out.println("插入20000条数据，耗时(毫秒)："+(end-start));

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}finally{

//遵循：resultset-->statment-->connection这样的关闭顺序！一定要将三个trycatch块，分开写！

try {

if(rs!=null){

rs.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(stmt!=null){

stmt.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 第271集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季271集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC当中事务的概念：

        基本概念：

            一组要么同时执行成功，要么同时执行失败的sql语句，是数据库操作的一个执行单元。

        -事务开始于DML语句（insert ，update,delete）；

         -事务结束于：

         .执行commit或者rollback语句

         .执行一条DDL语句，

         .执行一条DCL语句。例如：grant语句后会自动commit语句。

         .断开与数据库的连接。

         .执行了一条DML语句，该语句却失败了，在这种情况中，会为这个无效的操作DML语句执行rollback语句。

# 事务的四大特点（ACID）；

    -atomicity(原子性);

    -consistency(一致性);

    -isolation（隔离性）；

    -durability（持久性）；

# 事务隔离级别从低到高：

    -读取未提交（Read Uncommitted）；

    -读取已提交（Read Committed）；

    -可重复读（Repeatable Read）；

    -序列化（Serializable）；

## 事务的有关代码操作如下：

package com.bjsxt.jdbc;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

 \* 测试事务的基本概念和用法

 \* @author 高淇 www.sxt.cn

 \*

 \*/

public class Demo06 {

public static void main(String[] args) {

Connection conn = null;

PreparedStatement ps1 = null;

PreparedStatement ps2 = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

conn.setAutoCommit(false); //JDBC中默认是true，自动提交事务

ps1 = conn.prepareStatement("insert into t\_user (username,pwd) values (?,?)");

ps1.setObject(1, "高淇");

ps1.setObject(2, "123456");

ps1.execute();

System.out.println("插入一个用户,高淇");

try {

Thread.sleep(6000);

} catch (InterruptedException e) {

e.printStackTrace();

}

ps2 = conn.prepareStatement("insert into t\_user (username,pwd) values (?,?,?)");

ps2.setObject(1, "马士兵");

ps2.setObject(2, "123456");

ps2.execute();

System.out.println("插入一个用户,马士兵");

conn.commit();

} catch (ClassNotFoundException e) {

e.printStackTrace();

try {

conn.rollback(); //回滚

} catch (SQLException e1) {

e1.printStackTrace();

}

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

if(ps1!=null){

ps1.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 

# 第272集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季272集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC技术当中的时间类型

## 1） java.util.Date

        -子类：java.sql.Date  表示年月日

        -子类：java.sql.Time 表示时分秒

        -子类：java.sql.Timestamp 表示年月日时分秒。

## 2）日期比较处理

        -插入随机日期

        -取出指定日期范围的记录

package com.bjsxt.jdbc;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.Timestamp;

import java.util.Random;

/\*\*

 \* 测试时间处理(java.sql.Date,Time,Timestamp)

 \* @author 高淇 www.sxt.cn

 \*/

public class Demo07 {

public static void main(String[] args) {

Connection conn = null;

PreparedStatement ps = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

for(int i=0;i<1000;i++){

ps = conn.prepareStatement("insert into t\_user (username,pwd,regTime,lastLoginTime) values (?,?,?,?)");

ps.setObject(1, "高淇"+i);

ps.setObject(2, "123456");

int rand =  100000000+new Random().nextInt(1000000000);

java.sql.Date date = new java.sql.Date(System.currentTimeMillis()-rand);

Timestamp stamp = new Timestamp(System.currentTimeMillis()-rand);  //如果需要插入指定日期，可以使用Calendar、DateFormat

ps.setDate(3, date);

ps.setTimestamp(4, stamp);

ps.execute();

}

System.out.println("插入一个用户,高淇");

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 第273集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季273集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC之时间类的应用

## 查某段时间的数据的实现：

        java.util.Date：

        java.sql.TimeStamp；

        java.sql.Time;

## 字符串--》时间类的转换：

        DateFormat format =new SimpleDateFromat("yyyy-MM-dd hh:mm:ss");

        long start=  format.parse(String str).getTime();

package com.bjsxt.jdbc;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.Timestamp;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Random;

/\*\*

 \* 测试时间处理(java.sql.Date,Time,Timestamp),取出指定时间段的数据

 \* @author 高淇 www.sxt.cn

 \*/

public class Demo08 {

/\*\*

 \* 将字符串代表的日期转为long数字(格式：yyyy-MM-dd hh:mm:ss)

 \* @param dateStr

 \* @return

 \*/

public static  long  str2Date(String dateStr){

DateFormat format = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");

try {

return format.parse(dateStr).getTime();

} catch (ParseException e) {

e.printStackTrace();

return 0;

}

}

public static void main(String[] args) {

Connection conn = null;

PreparedStatement ps = null;

ResultSet rs = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

// ps = conn.prepareStatement("select \* from t\_user where regTime>? and regTime<?");

// java.sql.Date start = new java.sql.Date(str2Date("2015-4-10 10:23:45"));

// java.sql.Date end = new java.sql.Date(str2Date("2015-4-13 10:23:45"));

// ps.setObject(1, start);

// ps.setObject(2, end);

ps = conn.prepareStatement("select \* from t\_user where lastLoginTime>? and lastLoginTime<?  order by lastLoginTime ");

Timestamp start = new Timestamp(str2Date("2015-4-18 8:10:20"));

Timestamp end = new Timestamp(str2Date("2015-4-18  9:9:10"));

ps.setObject(1, start);

ps.setObject(2, end);

rs = ps.executeQuery();

while(rs.next()){

System.out.println(rs.getInt("id")+"--"+rs.getString("username")+"--"+rs.getTimestamp("lastLoginTime"));

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 第274集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季274集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC之Clob（Charactor Large Object）文本存储大数据对象：

## -CLOB用于存储大量的文本数据

    -大字段有些特殊，不同数据库处理的方式不一样，大字段的操作常常是以流的方式来处理的，而非一般的字段，一次即可读出数据。

## Mysql中相关类型如下：

        -TINYTEXT 最大长度为255字符的text列

        -TEXT【M】最大长度为65563字符的text列

        -MEDIMTEXT最大长度为16777215字符的列

        -LONGTEXT最大长度为4294,967,2956或者4GB字符的列；

## 代码展示如下：

package com.bjsxt.jdbc;

import java.io.Reader;

import java.sql.Clob;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

/\*\*

 \* 测试CLOB  文本大对象的使用

 \* 包含：将字符串、文件内容插入数据库中的CLOB字段、将CLOB字段值取出来的操作。

 \* @author 高淇 www.sxt.cn

 \*

 \*/

public class Demo09 {

public static void main(String[] args) {

Connection conn = null;

PreparedStatement ps = null;

ResultSet rs = null;

Reader r  = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

// ps = conn.prepareStatement("insert into t\_user (username,myInfo) values (?,?) ");

// ps.setString(1, "高淇");

// ps.setClob(2, new FileReader(new File("d:/a.txt")));  //将文本文件内容直接输入到数据库中

//将程序中的字符串输入到数据库的CLOB字段中

// ps.setClob(2, new BufferedReader(new InputStreamReader(new ByteArrayInputStream("aaaabbbbbb".getBytes()))));

ps = conn.prepareStatement("select \* from t\_user where id=?");

ps.setObject(1, 101024);

rs = ps.executeQuery();

while(rs.next()){

Clob c = rs.getClob("myInfo");

r  = c.getCharacterStream();

int temp = 0;

while((temp=r.read())!=-1){

System.out.print((char)temp);

}

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(r!=null){

r.close();

}

} catch (Exception e) {

e.printStackTrace();

}

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

# 第275集

尚学堂JAVA最全教程

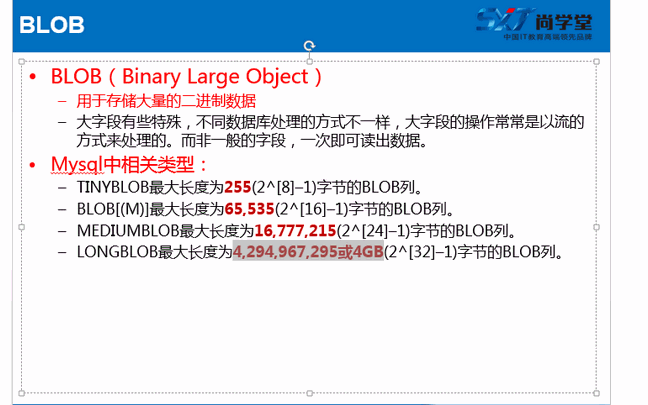
配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季275集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC之Bloc对象的使用：



# 关于Blob的代码演示如下：

package com.bjsxt.jdbc;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.io.OutputStream;

import java.sql.Blob;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

/\*\*

 \* 测试BLOB  二进制大对象的使用

 \* @author 高淇 www.sxt.cn

 \*/

public class Demo10 {

public static void main(String[] args) {

Connection conn = null;

PreparedStatement ps = null;

ResultSet rs = null;

InputStream is  = null;

OutputStream os = null;

try {

//加载驱动类

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testjdbc","root","123456");

// ps = conn.prepareStatement("insert into t\_user (username,headImg) values (?,?) ");

// ps.setString(1, "高淇");

// ps.setBlob(2, new FileInputStream("d:/icon.jpg"));

// ps.execute();

ps = conn.prepareStatement("select \* from t\_user where id=?");

ps.setObject(1, 101026);

rs = ps.executeQuery();

while(rs.next()){

Blob b = rs.getBlob("headImg");

is  = b.getBinaryStream();

os = new FileOutputStream("d:/a.jpg");

int temp = 0;

while((temp=is.read())!=-1){

os.write(temp);

}

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}finally{

try {

if(is!=null){

is.close();

}

} catch (Exception e) {

e.printStackTrace();

}

try {

if(os!=null){

os.close();

}

} catch (Exception e) {

e.printStackTrace();

}

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

代码中的图片，以及本地的资源大家可以观看视频，自己到网上下载一个即可：希望大家能够举一反三。

# 第276集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季276集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC的总结以封装：

## 配置文件的代码为：

mysqlDriver=com.mysql.jdbc.Driver

mysqlURL=jdbc\:mysql\://localhost\:3306/testjdbc

mysqlUser=root

mysqlPwd=123456

oracleDriver=oracle.jdbc.driver.OracleDriver

oracleURL=jdbc\:oracle\:thin\:@localhost\:1521\:orcl

oracleUser=scott

oraclePwd=tiger

## JDBCUtil工具类的代码：

package com.bjsxt.jdbc;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Properties;

public class JDBCUtil {

static Properties pros = null;   //可以帮助读取和处理资源文件中的信息

static {   //加载JDBCUtil类的时候调用

pros = new Properties();

try {

pros.load(Thread.currentThread().getContextClassLoader().getResourceAsStream("db.properties"));

} catch (IOException e) {

e.printStackTrace();

}

}

public static Connection getMysqlConn(){

try {

Class.forName(pros.getProperty("mysqlDriver"));

return DriverManager.getConnection(pros.getProperty("mysqlURL"),

pros.getProperty("mysqlUser"),pros.getProperty("mysqlPwd"));

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

public static Connection getOracleConn(){

try {

Class.forName(pros.getProperty("oracleDriver"));

return DriverManager.getConnection(pros.getProperty("oracleURL"),

pros.getProperty("oracleUser"),pros.getProperty("oraclePwd"));

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

public static void close(ResultSet rs,Statement ps,Connection conn){

try {

if(rs!=null){

rs.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void close(Statement ps,Connection conn){

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void close(Connection conn){

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

# 第277集

尚学堂JAVA最全教程

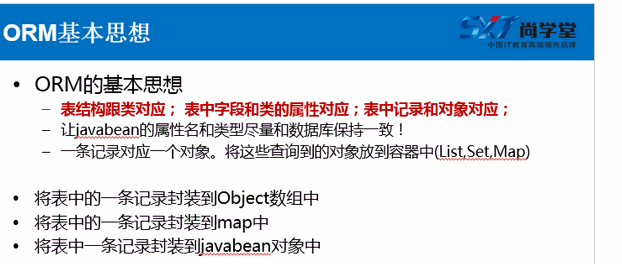
配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季277集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# ORM（Object Relationship Mapping）的基本思想：



# 代码的展示

package com.bjsxt.testorm;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

/\*\*\*

 \* 使用Object[]来封装一条记录

 \* 使用List<Object[]>存储多条记录

 \* @author Administrator

 \*

 \*/

public class Demo01 {

public static void main(String[] args) {

Connection conn = JDBCUtil.getMysqlConn();

PreparedStatement ps = null;

ResultSet rs = null;

List<Object[]> list= new ArrayList<Object[]>();

try {

ps = conn.prepareStatement("select empname,salary,age from emp where id>?");

ps.setObject(1, 1);

rs = ps.executeQuery();

while(rs.next()){

Object[] objs= new Object[3];   //一个Object数组封装了一条记录的信息！

// System.out.println(rs.getString(1)+"--"+rs.getDouble(2)+"--"+rs.getInt(3));

objs[0]= rs.getString(1);

objs[1] = rs.getObject(2);

objs[2] = rs.getObject(3);

list.add(objs);

}

} catch (SQLException e) {

e.printStackTrace();

}finally{

JDBCUtil.close(rs, ps, conn);

}

for(Object[] objs:list){

System.out.println(""+objs[0]+objs[1]+objs[2]);

}

}

}

# 第278集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季278集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC技术的应用：

   1） 将表中的一条记录封装到Object数组中。

    2）将表中的一条记录封装到Map中

    3）将表中一条记录封装到javabean中。

## 封装到map中的代码演示如下：

package com.bjsxt.testorm;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

/\*\*\*

 \* 使用Map来封装一条记录

 \* 使用List<Map>,Map<Map>存储多条记录

 \* @author Administrator

 \*

 \*/

public class Demo02 {

public static void test01(){

Connection conn = JDBCUtil.getMysqlConn();

PreparedStatement ps = null;

ResultSet rs = null;

Map<String,Object> row = new HashMap<String, Object>(); //使用一个Map封装一条记录

try {

ps = conn.prepareStatement("select empname,salary,age from emp where id=?");

ps.setObject(1, 1);

rs = ps.executeQuery();

while(rs.next()){

// System.out.println(rs.getString(1)+"--"+rs.getDouble(2)+"--"+rs.getInt(3));

row.put("empname", rs.getObject(1));

row.put("salary", rs.getObject(2));

row.put("age", rs.getObject(3));

}

} catch (SQLException e) {

e.printStackTrace();

}finally{

JDBCUtil.close(rs, ps, conn);

}

//遍历Map，就是遍历这一行的多列的信息

for(String key:row.keySet()){

System.out.print(key+"--"+row.get(key)+"\t");

}

}

public static void test02(){

Connection conn = JDBCUtil.getMysqlConn();

PreparedStatement ps = null;

ResultSet rs = null;

List<Map<String,Object>> list= new ArrayList<Map<String,Object>>();

try {

ps = conn.prepareStatement("select empname,salary,age from emp where id>?");

ps.setObject(1, 1);

rs = ps.executeQuery();

while(rs.next()){

// System.out.println(rs.getString(1)+"--"+rs.getDouble(2)+"--"+rs.getInt(3));

Map<String,Object> row = new HashMap<String, Object>(); //使用一个Map封装一条记录

row.put("empname", rs.getObject(1));

row.put("salary", rs.getObject(2));

row.put("age", rs.getObject(3));

list.add(row);

}

} catch (SQLException e) {

e.printStackTrace();

}finally{

JDBCUtil.close(rs, ps, conn);

}

//遍历Map，就是遍历这一行的多列的信息

for(Map<String,Object> row:list){

for(String key:row.keySet()){

System.out.print(key+"--"+row.get(key)+"\t");

}

System.out.println();

}

}

public static void test03(){

Connection conn = JDBCUtil.getMysqlConn();

PreparedStatement ps = null;

ResultSet rs = null;

Map<String,Map<String,Object>> maps = new HashMap<String,Map<String,Object>>();

try {

ps = conn.prepareStatement("select empname,salary,age from emp where id>?");

ps.setObject(1, 1);

rs = ps.executeQuery();

while(rs.next()){

// System.out.println(rs.getString(1)+"--"+rs.getDouble(2)+"--"+rs.getInt(3));

Map<String,Object> row = new HashMap<String, Object>(); //使用一个Map封装一条记录

row.put("empname", rs.getObject(1));

row.put("salary", rs.getObject(2));

row.put("age", rs.getObject(3));

maps.put(rs.getString(1), row);

}

} catch (SQLException e) {

e.printStackTrace();

}finally{

JDBCUtil.close(rs, ps, conn);

}

//遍历Map，就是遍历这一行的多列的信息

for(String empname:maps.keySet()){

Map<String,Object> row = maps.get(empname);

for(String key:row.keySet()){

System.out.print(key+"--"+row.get(key)+"\t");

}

System.out.println();

}

}

public static void main(String[] args) {

test03();

}

}

# 第279集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季279集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# JDBC技术之将一条记录封装的javaBean中。

    数据库中建立了两张表分别为：emp(雇员表),dept(部门表)：

    我们封装两个类分别与其对应：

## Emp类的封装代码如下：

package com.bjsxt.testorm;

import java.sql.Date;

public class Emp {   //表结构和类对应

private Integer id;

private String empname;

private Integer age;

private Double salary;

private Date birthday;

private Integer deptId;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getEmpname() {

return empname;

}

public void setEmpname(String empname) {

this.empname = empname;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

public Double getSalary() {

return salary;

}

public void setSalary(Double salary) {

this.salary = salary;

}

public Date getBirthday() {

return birthday;

}

public void setBirthday(Date birthday) {

this.birthday = birthday;

}

public Integer getDeptId() {

return deptId;

}

public void setDeptId(Integer deptId) {

this.deptId = deptId;

}

public Emp(Integer id, String empname, Integer age, Double salary,

Date birthday, Integer deptId) {

super();

this.id = id;

this.empname = empname;

this.age = age;

this.salary = salary;

this.birthday = birthday;

this.deptId = deptId;

}

public Emp(String empname, Integer age, Double salary, Date birthday,

Integer deptId) {

super();

this.empname = empname;

this.age = age;

this.salary = salary;

this.birthday = birthday;

this.deptId = deptId;

}

public Emp(String empname,Double salary, Integer age) {

super();

this.empname = empname;

this.age = age;

this.salary = salary;

}

public Emp() {

}

}

## javaBean Dept类的封装如下：

package com.bjsxt.testorm;

public class Dept {

private Integer id;

private String dname;

private String address;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getDname() {

return dname;

}

public void setDname(String dname) {

this.dname = dname;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public Dept(Integer id, String dname, String address) {

super();

this.id = id;

this.dname = dname;

this.address = address;

}

public Dept(String dname, String address) {

super();

this.dname = dname;

this.address = address;

}

public Dept() {

}

}

## 测试代码如下：

package com.bjsxt.testorm;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

/\*\*\*

 \* 使用Javabean对象来封装一条记录

 \* 使用List<Javabean>存储多条记录

 \* @author Administrator

 \*

 \*/

public class Demo03 {

public static void test01(){

Connection conn = JDBCUtil.getMysqlConn();

PreparedStatement ps = null;

ResultSet rs = null;

Emp emp = null;

try {

ps = conn.prepareStatement("select empname,salary,age from emp where id=?");

ps.setObject(1, 1);

rs = ps.executeQuery();

while(rs.next()){

// System.out.println(rs.getString(1)+"--"+rs.getDouble(2)+"--"+rs.getInt(3));

emp = new Emp(rs.getString(1),rs.getDouble(2),rs.getInt(3));

}

} catch (SQLException e) {

e.printStackTrace();

}finally{

JDBCUtil.close(rs, ps, conn);

}

System.out.println(emp.getEmpname()+"-"+emp.getSalary()+"-"+emp.getAge());

}

public static void test02(){

Connection conn = JDBCUtil.getMysqlConn();

PreparedStatement ps = null;

ResultSet rs = null;

List<Emp> list= new ArrayList<Emp>();

try {

ps = conn.prepareStatement("select empname,salary,age from emp where id>?");

ps.setObject(1, 1);

rs = ps.executeQuery();

while(rs.next()){

Emp emp = new Emp(rs.getString(1),rs.getDouble(2),rs.getInt(3));

list.add(emp);

}

} catch (SQLException e) {

e.printStackTrace();

}finally{

JDBCUtil.close(rs, ps, conn);

}

//遍历List，就是遍历这一行的多列的信息

for(Emp emp:list){

System.out.println(emp.getEmpname()+"-"+emp.getSalary()+"-"+emp.getAge());

}

}

public static void main(String[] args) {

test02();

}

}

# 第280集

尚学堂JAVA最全教程

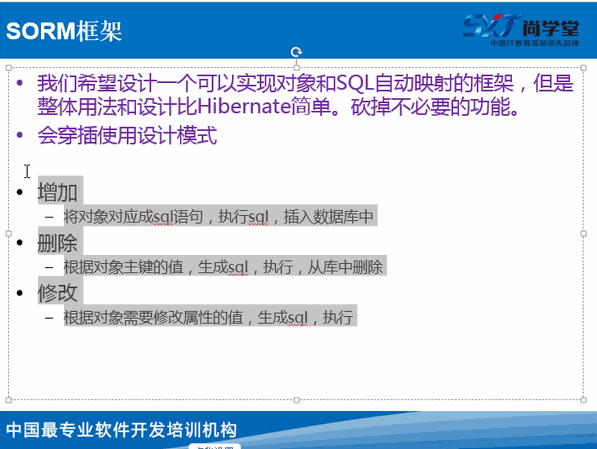
配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季280集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

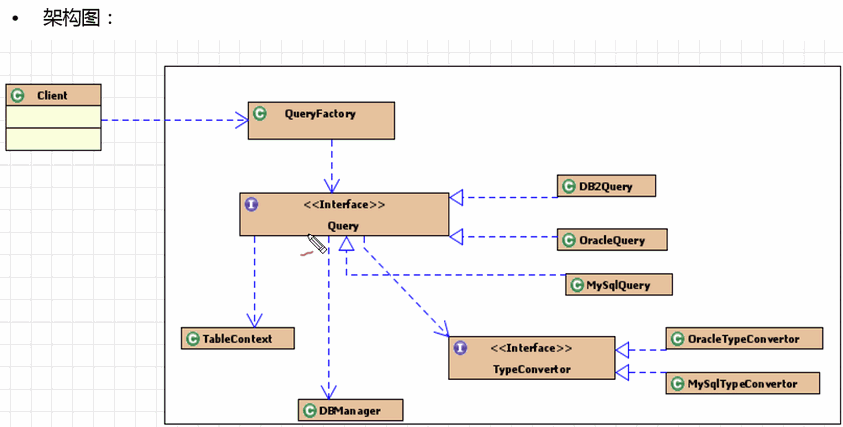
【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

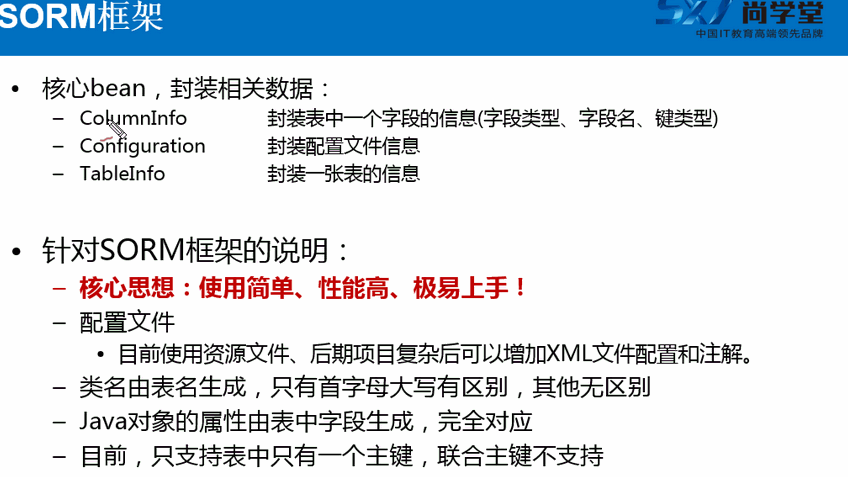
# SORM（simple Object relationship Mapping）的基本思想及框架的结构：



## SORM的架构







# 第281-282集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季281-282集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 对SORM进行规划设计：

        com.bjsxt.sorm.core

        com.bjsxt.bean

        com.bjsxt.utils

        com.bjsxt.po

        src下建立一个readme.txt文档

        所有类的都是搭建一个框架没有什么操作的代码：

# 这两集主要是封装了Configuration类和TableContext类：

package com.bjsxt.sorm.bean;

/\*\*

 \* 管理配置信息

 \* @author gaoqi www.sxt.cn

 \*

 \*/

public class Configuration {

/\*\*

 \* 驱动类

 \*/

private String driver;

/\*\*

 \* jdbc的url

 \*/

private String url;

/\*\*

 \* 数据库的用户名

 \*/

private String user;

/\*\*

 \* 数据库的密码

 \*/

private String pwd;

/\*\*

 \* 正在使用哪个数据库

 \*/

private String usingDB;

/\*\*

 \* 项目的源码路径

 \*/

private String srcPath;

/\*\*

 \* 扫描生成java类的包(po的意思是：Persistence object持久化对象)

 \*/

private String poPackage;

public Configuration() {

}

public Configuration(String driver, String url, String user, String pwd,

String usingDB, String srcPath, String poPackage) {

super();

this.driver = driver;

this.url = url;

this.user = user;

this.pwd = pwd;

this.usingDB = usingDB;

this.srcPath = srcPath;

this.poPackage = poPackage;

}

public String getDriver() {

return driver;

}

public void setDriver(String driver) {

this.driver = driver;

}

public String getUrl() {

return url;

}

public void setUrl(String url) {

this.url = url;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getPwd() {

return pwd;

}

public void setPwd(String pwd) {

this.pwd = pwd;

}

public String getUsingDB() {

return usingDB;

}

public void setUsingDB(String usingDB) {

this.usingDB = usingDB;

}

public String getSrcPath() {

return srcPath;

}

public void setSrcPath(String srcPath) {

this.srcPath = srcPath;

}

public String getPoPackage() {

return poPackage;

}

public void setPoPackage(String poPackage) {

this.poPackage = poPackage;

}

}

## TableContext类：

package com.bjsxt.sorm.core;

import java.sql.Connection;

import java.sql.DatabaseMetaData;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import com.bjsxt.sorm.bean.ColumnInfo;

import com.bjsxt.sorm.bean.TableInfo;

import com.bjsxt.sorm.utils.JavaFileUtils;

import com.bjsxt.sorm.utils.StringUtils;

/\*\*

 \* 负责获取管理数据库所有表结构和类结构的关系，并可以根据表结构生成类结构。

 \* @author gaoqi www.sxt.cn

 \*

 \*/

public class TableContext {

/\*\*

 \* 表名为key，表信息对象为value

 \*/

public static  Map<String,TableInfo>  tables = new HashMap<String,TableInfo>();

/\*\*

 \* 将po的class对象和表信息对象关联起来，便于重用！

 \*/

public static  Map<Class,TableInfo>  poClassTableMap = new HashMap<Class,TableInfo>();

private TableContext(){

}

static {

try {

//初始化获得表的信息

Connection con = DBManager.getConn();

DatabaseMetaData dbmd = con.getMetaData();

ResultSet tableRet = dbmd.getTables(null, "%","%",new String[]{"TABLE"});

while(tableRet.next()){

String tableName = (String) tableRet.getObject("TABLE\_NAME");

TableInfo ti = new TableInfo(tableName, new ArrayList<ColumnInfo>()

,new HashMap<String, ColumnInfo>());

tables.put(tableName, ti);

ResultSet set = dbmd.getColumns(null, "%", tableName, "%");  //查询表中的所有字段

while(set.next()){

ColumnInfo ci = new ColumnInfo(set.getString("COLUMN\_NAME"),

set.getString("TYPE\_NAME"), 0);

ti.getColumns().put(set.getString("COLUMN\_NAME"), ci);

}

ResultSet set2 = dbmd.getPrimaryKeys(null, "%", tableName);  //查询t\_user表中的主键

while(set2.next()){

ColumnInfo ci2 = (ColumnInfo) ti.getColumns().get(set2.getObject("COLUMN\_NAME"));

ci2.setKeyType(1);  //设置为主键类型

ti.getPriKeys().add(ci2);

}

if(ti.getPriKeys().size()>0){  //取唯一主键。。方便使用。如果是联合主键。则为空！

ti.setOnlyPriKey(ti.getPriKeys().get(0));

}

}

} catch (SQLException e) {

e.printStackTrace();

}

//更新类结构

updateJavaPOFile();

//加载po包下面所有的类，便于重用，提高效率！

loadPOTables();

}

/\*\*

 \* 根据表结构，更新配置的po包下面的java类

 \* 实现了从表结构转化到类结构

 \*/

public static void updateJavaPOFile(){

Map<String,TableInfo> map = TableContext.tables;

for(TableInfo t:map.values()){

JavaFileUtils.createJavaPOFile(t,new MySqlTypeConvertor());

}

}

/\*\*

 \* 加载po包下面的类

 \*/

public static void loadPOTables(){

for(TableInfo tableInfo:tables.values()){

try {

Class c = Class.forName(DBManager.getConf().getPoPackage()

+"."+StringUtils.firstChar2UpperCase(tableInfo.getTname()));

poClassTableMap.put(c, tableInfo);

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

}

public static void main(String[] args) {

 Map<String,TableInfo>  tables = TableContext.tables;

 System.out.println(tables);

}

}

## DBManager类代码如下：

package com.bjsxt.sorm.core;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Properties;

import com.bjsxt.sorm.bean.Configuration;

/\*\*

 \* 根据配置信息，维持连接对象的管理(增加连接池功能)

 \* @author Administrator

 \*

 \*/

public class DBManager {

private static Configuration conf;

static {  //静态代码块

Properties pros = new Properties();

try {

pros.load(Thread.currentThread().getContextClassLoader().getResourceAsStream("db.properties"));

} catch (IOException e) {

e.printStackTrace();

}

conf = new Configuration();

conf.setDriver(pros.getProperty("driver"));

conf.setPoPackage(pros.getProperty("poPackage"));

conf.setPwd(pros.getProperty("pwd"));

conf.setSrcPath(pros.getProperty("srcPath"));

conf.setUrl(pros.getProperty("url"));

conf.setUser(pros.getProperty("user"));

conf.setUsingDB(pros.getProperty("usingDB"));

}

public static Connection getConn(){

try {

Class.forName(conf.getDriver());

return DriverManager.getConnection(conf.getUrl(),

conf.getUser(),conf.getPwd());     //直接建立连接，后期增加连接池处理，提高效率！！！

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

public static void close(ResultSet rs,Statement ps,Connection conn){

try {

if(rs!=null){

rs.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void close(Statement ps,Connection conn){

try {

if(ps!=null){

ps.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void close(Connection conn){

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

/\*\*

 \* 返回Configuration对象

 \* @return

 \*/

public static Configuration getConf(){

return conf;

}

}

# 第283集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季283集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM的进阶：

    这集主要是讲解了mysql的数据类型如何转换成java的数据类型代码展示如下：

package com.bjsxt.sorm.core;

/\*\*

 \* mysql数据类型和java数据类型的转换

 \* @author gaoqi

 \*

 \*/

public class MySqlTypeConvertor implements TypeConvertor {

@Override

public String databaseType2JavaType(String columnType) {

//varchar-->String

if("varchar".equalsIgnoreCase(columnType)||"char".equalsIgnoreCase(columnType)){

return "String";

}else if("int".equalsIgnoreCase(columnType)

||"tinyint".equalsIgnoreCase(columnType)

||"smallint".equalsIgnoreCase(columnType)

||"integer".equalsIgnoreCase(columnType)

){

return "Integer";

}else if("bigint".equalsIgnoreCase(columnType)){

return "Long";

}else if("double".equalsIgnoreCase(columnType)||"float".equalsIgnoreCase(columnType)){

return "Double";

}else if("clob".equalsIgnoreCase(columnType)){

return "java.sql.CLob";

}else if("blob".equalsIgnoreCase(columnType)){

return "java.sql.BLob";

}else if("date".equalsIgnoreCase(columnType)){

return "java.sql.Date";

}else if("time".equalsIgnoreCase(columnType)){

return "java.sql.Time";

}else if("timestamp".equalsIgnoreCase(columnType)){

return "java.sql.Timestamp";

}

return null;

}

@Override

public String javaType2DatabaseType(String javaDataType) {

return null;

}

}

# 第284集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季284集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM进阶：

    这集主要封装了JavaFieldGetSet类和JavaFileUtils类，以及简单的测试：

    代码如下：

## JavaFieldGetSet类

package com.bjsxt.sorm.bean;

/\*\*

 \* 封装了java属性和get、set方法的源代码

 \* @author gaoqi

 \*

 \*/

public class JavaFieldGetSet {

/\*\*

 \* 属性的源码信息。如：private int userId;

 \*/

private String fieldInfo;

/\*\*

 \* get方法的源码信息.如：public int getUserId(){}

 \*/

private String getInfo;

/\*\*

 \* set方法的源码信息.如：public void setUserId(int id){this.id = id;}

 \*/

private String setInfo;

@Override

public String toString() {

System.out.println(fieldInfo);

System.out.println(getInfo);

System.out.println(setInfo);

return super.toString();

}

public String getFieldInfo() {

return fieldInfo;

}

public void setFieldInfo(String fieldInfo) {

this.fieldInfo = fieldInfo;

}

public String getGetInfo() {

return getInfo;

}

public void setGetInfo(String getInfo) {

this.getInfo = getInfo;

}

public String getSetInfo() {

return setInfo;

}

public void setSetInfo(String setInfo) {

this.setInfo = setInfo;

}

public JavaFieldGetSet(String fieldInfo, String getInfo, String setInfo) {

super();

this.fieldInfo = fieldInfo;

this.getInfo = getInfo;

this.setInfo = setInfo;

}

public JavaFieldGetSet() {

}

}

## JavaFileUtils类代码

package com.bjsxt.sorm.utils;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import com.bjsxt.sorm.bean.ColumnInfo;

import com.bjsxt.sorm.bean.JavaFieldGetSet;

import com.bjsxt.sorm.bean.TableInfo;

import com.bjsxt.sorm.core.DBManager;

import com.bjsxt.sorm.core.MySqlTypeConvertor;

import com.bjsxt.sorm.core.TableContext;

import com.bjsxt.sorm.core.TypeConvertor;

/\*\*

 \* 封装了生成Java文件(源代码)常用的操作

 \* @author gaoqi www.sxt.cn

 \*

 \*/

public class JavaFileUtils {

/\*\*

 \* 根据字段信息生成java属性信息。如：varchar username-->private String username;以及相应的set和get方法源码

 \* @param column 字段信息

 \* @param convertor 类型转化器

 \* @return java属性和set/get方法源码

 \*/

public static JavaFieldGetSet createFieldGetSetSRC(ColumnInfo column,TypeConvertor convertor){

JavaFieldGetSet jfgs  = new JavaFieldGetSet();

String javaFieldType = convertor.databaseType2JavaType(column.getDataType());

jfgs.setFieldInfo("\tprivate "+javaFieldType+" "+column.getName()+";\n");

//public String getUsername(){return username;}

//生成get方法的源代码

StringBuilder getSrc = new StringBuilder();

getSrc.append("\tpublic "+javaFieldType+" get"+StringUtils.firstChar2UpperCase(column.getName())+"(){\n");

getSrc.append("\t\treturn "+column.getName()+";\n");

getSrc.append("\t}\n");

jfgs.setGetInfo(getSrc.toString());

//public void setUsername(String username){this.username=username;}

//生成set方法的源代码

StringBuilder setSrc = new StringBuilder();

setSrc.append("\tpublic void set"+StringUtils.firstChar2UpperCase(column.getName())+"(");

setSrc.append(javaFieldType+" "+column.getName()+"){\n");

setSrc.append("\t\tthis."+column.getName()+"="+column.getName()+";\n");

setSrc.append("\t}\n");

jfgs.setSetInfo(setSrc.toString());

return jfgs;

}

/\*\*

 \* 根据表信息生成java类的源代码

 \* @param tableInfo 表信息

 \* @param convertor 数据类型转化器

 \* @return java类的源代码

 \*/

public static String createJavaSrc(TableInfo tableInfo,TypeConvertor convertor){

Map<String,ColumnInfo> columns = tableInfo.getColumns();

List<JavaFieldGetSet> javaFields = new ArrayList<JavaFieldGetSet>();

for(ColumnInfo c:columns.values()){

javaFields.add(createFieldGetSetSRC(c,convertor));

}

StringBuilder src = new StringBuilder();

//生成package语句

src.append("package "+DBManager.getConf().getPoPackage()+";\n\n");

//生成import语句

src.append("import java.sql.\*;\n");

src.append("import java.util.\*;\n\n");

//生成类声明语句

src.append("public class "+StringUtils.firstChar2UpperCase(tableInfo.getTname())+" {\n\n");

//生成属性列表

for(JavaFieldGetSet f:javaFields){

src.append(f.getFieldInfo());

}

src.append("\n\n");

//生成get方法列表

for(JavaFieldGetSet f:javaFields){

src.append(f.getGetInfo());

}

//生成set方法列表

for(JavaFieldGetSet f:javaFields){

src.append(f.getSetInfo());

}

//生成类结束

src.append("}\n");

return src.toString();

}

public static void createJavaPOFile(TableInfo tableInfo,TypeConvertor convertor){

String src = createJavaSrc(tableInfo,convertor);

String srcPath = DBManager.getConf().getSrcPath()+"\\";

String packagePath = DBManager.getConf().getPoPackage().replaceAll("\\.", "/");

File f = new File(srcPath+packagePath);

if(!f.exists()){  //如果指定目录不存在，则帮助用户建立

f.mkdirs();

}

BufferedWriter bw = null;

try {

bw = new BufferedWriter(new FileWriter(f.getAbsoluteFile()+"/"+StringUtils.firstChar2UpperCase(tableInfo.getTname())+".java"));

bw.write(src);

System.out.println("建立表"+tableInfo.getTname()+

"对应的java类："+StringUtils.firstChar2UpperCase(tableInfo.getTname())+".java");

} catch (IOException e) {

e.printStackTrace();

}finally{

try {

if(bw!=null){

bw.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

public static void main(String[] args) {

// ColumnInfo ci = new ColumnInfo("id", "int", 0);

// JavaFieldGetSet f = createFieldGetSetSRC(ci,new MySqlTypeConvertor());

// System.out.println(f);

Map<String,TableInfo> map = TableContext.tables;

for(TableInfo t:map.values()){

createJavaPOFile(t,new MySqlTypeConvertor());

}

}

}

# 第286集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季286集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM进阶：

    本集还是对JavaFileUtils类方法的封装：

package com.bjsxt.sorm.utils;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import com.bjsxt.sorm.bean.ColumnInfo;

import com.bjsxt.sorm.bean.JavaFieldGetSet;

import com.bjsxt.sorm.bean.TableInfo;

import com.bjsxt.sorm.core.DBManager;

import com.bjsxt.sorm.core.MySqlTypeConvertor;

import com.bjsxt.sorm.core.TableContext;

import com.bjsxt.sorm.core.TypeConvertor;

/\*\*

 \* 封装了生成Java文件(源代码)常用的操作

 \* @author gaoqi www.sxt.cn

 \*

 \*/

public class JavaFileUtils {

/\*\*

 \* 根据字段信息生成java属性信息。如：varchar username-->private String username;以及相应的set和get方法源码

 \* @param column 字段信息

 \* @param convertor 类型转化器

 \* @return java属性和set/get方法源码

 \*/

public static JavaFieldGetSet createFieldGetSetSRC(ColumnInfo column,TypeConvertor convertor){

JavaFieldGetSet jfgs  = new JavaFieldGetSet();

String javaFieldType = convertor.databaseType2JavaType(column.getDataType());

jfgs.setFieldInfo("\tprivate "+javaFieldType+" "+column.getName()+";\n");

//public String getUsername(){return username;}

//生成get方法的源代码

StringBuilder getSrc = new StringBuilder();

getSrc.append("\tpublic "+javaFieldType+" get"+StringUtils.firstChar2UpperCase(column.getName())+"(){\n");

getSrc.append("\t\treturn "+column.getName()+";\n");

getSrc.append("\t}\n");

jfgs.setGetInfo(getSrc.toString());

//public void setUsername(String username){this.username=username;}

//生成set方法的源代码

StringBuilder setSrc = new StringBuilder();

setSrc.append("\tpublic void set"+StringUtils.firstChar2UpperCase(column.getName())+"(");

setSrc.append(javaFieldType+" "+column.getName()+"){\n");

setSrc.append("\t\tthis."+column.getName()+"="+column.getName()+";\n");

setSrc.append("\t}\n");

jfgs.setSetInfo(setSrc.toString());

return jfgs;

}

/\*\*

 \* 根据表信息生成java类的源代码

 \* @param tableInfo 表信息

 \* @param convertor 数据类型转化器

 \* @return java类的源代码

 \*/

public static String createJavaSrc(TableInfo tableInfo,TypeConvertor convertor){

Map<String,ColumnInfo> columns = tableInfo.getColumns();

List<JavaFieldGetSet> javaFields = new ArrayList<JavaFieldGetSet>();

for(ColumnInfo c:columns.values()){

javaFields.add(createFieldGetSetSRC(c,convertor));

}

StringBuilder src = new StringBuilder();

//生成package语句

src.append("package "+DBManager.getConf().getPoPackage()+";\n\n");

//生成import语句

src.append("import java.sql.\*;\n");

src.append("import java.util.\*;\n\n");

//生成类声明语句

src.append("public class "+StringUtils.firstChar2UpperCase(tableInfo.getTname())+" {\n\n");

//生成属性列表

for(JavaFieldGetSet f:javaFields){

src.append(f.getFieldInfo());

}

src.append("\n\n");

//生成get方法列表

for(JavaFieldGetSet f:javaFields){

src.append(f.getGetInfo());

}

//生成set方法列表

for(JavaFieldGetSet f:javaFields){

src.append(f.getSetInfo());

}

//生成类结束

src.append("}\n");

return src.toString();

}

public static void createJavaPOFile(TableInfo tableInfo,TypeConvertor convertor){

String src = createJavaSrc(tableInfo,convertor);

String srcPath = DBManager.getConf().getSrcPath()+"\\";

String packagePath = DBManager.getConf().getPoPackage().replaceAll("\\.", "/");

File f = new File(srcPath+packagePath);

if(!f.exists()){  //如果指定目录不存在，则帮助用户建立

f.mkdirs();

}

BufferedWriter bw = null;

try {

bw = new BufferedWriter(new FileWriter(f.getAbsoluteFile()+"/"+StringUtils.firstChar2UpperCase(tableInfo.getTname())+".java"));

bw.write(src);

System.out.println("建立表"+tableInfo.getTname()+

"对应的java类："+StringUtils.firstChar2UpperCase(tableInfo.getTname())+".java");

} catch (IOException e) {

e.printStackTrace();

}finally{

try {

if(bw!=null){

bw.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

public static void main(String[] args) {

// ColumnInfo ci = new ColumnInfo("id", "int", 0);

// JavaFieldGetSet f = createFieldGetSetSRC(ci,new MySqlTypeConvertor());

// System.out.println(f);

Map<String,TableInfo> map = TableContext.tables;

for(TableInfo t:map.values()){

createJavaPOFile(t,new MySqlTypeConvertor());

}

}

}

# 第287集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季287集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# ＳＯＲＭ查询Query接口的实现类的继续封装：

## mysqlQuery类的代码如下：

package com.bjsxt.sorm.core;

import java.lang.reflect.Field;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.util.ArrayList;

import java.util.List;

import com.bjsxt.po.Emp;

import com.bjsxt.sorm.bean.ColumnInfo;

import com.bjsxt.sorm.bean.TableInfo;

import com.bjsxt.sorm.utils.JDBCUtils;

import com.bjsxt.sorm.utils.ReflectUtils;

import com.bjsxt.vo.EmpVO;

/\*\*

 \* 负责针对Mysql数据库的查询

 \* @author gaoqi

 \*

 \*/

public class MySqlQuery implements Query {

public static void testDML(){

Emp e = new Emp();

e.setEmpname("lily");

e.setBirthday(new java.sql.Date(System.currentTimeMillis()));

e.setAge(30);

e.setSalary(3000.8);

e.setId(1);

// new MySqlQuery().delete(e);

// new MySqlQuery().insert(e);

new MySqlQuery().update(e,new String[]{"empname","age","salary"});

}

public static void testQueryRows(){

List<Emp> list = new MySqlQuery().queryRows("select id,empname,age from emp where age>? and salary<?",

Emp.class, new Object[]{10,5000});

for(Emp e:list){

System.out.println(e.getEmpname());

}

String sql2 = "select e.id,e.empname,salary+bonus 'xinshui',age,d.dname 'deptName',d.address 'deptAddr' from emp e "

+"join dept d on e.deptId=d.id ";

List<EmpVO> list2 = new MySqlQuery().queryRows(sql2,

EmpVO.class, null);

for(EmpVO e:list2){

System.out.println(e.getEmpname()+"-"+e.getDeptAddr()+"-"+e.getXinshui());

}

}

public static void main(String[] args) {

// Number obj = (Number)new MySqlQuery().queryValue("select count(\*) from emp where salary>?",new Object[]{1000});

Number obj = new MySqlQuery().queryNumber("select count(\*) from emp where salary>?",new Object[]{1000});

System.out.println(obj.doubleValue());

}

@Override

public void delete(Class clazz, Object id) {

//Emp.class,2-->delete from emp where id=2

//通过Class对象找TableInfo

TableInfo tableInfo = TableContext.poClassTableMap.get(clazz);

//获得主键

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();

String sql = "delete from "+tableInfo.getTname()+" where "+onlyPriKey.getName()+"=? ";

executeDML(sql, new Object[]{id});

}

@Override

public void delete(Object obj) {

Class c = obj.getClass();

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();  //主键

//通过反射机制，调用属性对应的get方法或set方法

Object priKeyValue = ReflectUtils.invokeGet(onlyPriKey.getName(), obj);

delete(c, priKeyValue);

}

@Override

public int executeDML(String sql, Object[] params) {

Connection conn = DBManager.getConn();

int count = 0;

PreparedStatement ps = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

count  = ps.executeUpdate();

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return count;

}

@Override

public void insert(Object obj) {

//obj-->表中。             insert into 表名  (id,uname,pwd) values (?,?,?)

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

StringBuilder sql  = new StringBuilder("insert into "+tableInfo.getTname()+" (");

int countNotNullField = 0;   //计算不为null的属性值

Field[] fs = c.getDeclaredFields();

for(Field f:fs){

String fieldName = f.getName();

Object fieldValue = ReflectUtils.invokeGet(fieldName, obj);

if(fieldValue!=null){

countNotNullField++;

sql.append(fieldName+",");

params.add(fieldValue);

}

}

sql.setCharAt(sql.length()-1, ')');

sql.append(" values (");

for(int i=0;i<countNotNullField;i++){

sql.append("?,");

}

sql.setCharAt(sql.length()-1, ')');

executeDML(sql.toString(), params.toArray());

}

@Override

public Number queryNumber(String sql, Object[] params) {

return (Number)queryValue(sql, params);

}

@Override

public List queryRows(String sql, Class clazz, Object[] params) {

Connection conn = DBManager.getConn();

List list = null;    //存储查询结果的容器

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

ResultSetMetaData metaData = rs.getMetaData();

//多行

while(rs.next()){

if(list==null){

list = new ArrayList();

}

Object rowObj = clazz.newInstance();   //调用javabean的无参构造器

//多列       select username ,pwd,age from user where id>? and age>18

for(int i=0;i<metaData.getColumnCount();i++){

String columnName = metaData.getColumnLabel(i+1);  //username

Object columnValue = rs.getObject(i+1);

//调用rowObj对象的setUsername(String uname)方法，将columnValue的值设置进去

ReflectUtils.invokeSet(rowObj, columnName, columnValue);

}

list.add(rowObj);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return list;

}

@Override

public Object queryUniqueRow(String sql, Class clazz, Object[] params) {

List list = queryRows(sql, clazz, params);

return (list==null&&list.size()>0)?null:list.get(0);

}

@Override

public Object queryValue(String sql, Object[] params) {

Connection conn = DBManager.getConn();

Object value = null;    //存储查询结果的对象

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

while(rs.next()){

value = rs.getObject(1);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return value;

}

@Override

public int update(Object obj, String[] fieldNames) {

//obj{"uanme","pwd"}-->update 表名  set uname=?,pwd=? where id=?

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo  priKey = tableInfo.getOnlyPriKey();   //获得唯一的主键

StringBuilder sql  = new StringBuilder("update "+tableInfo.getTname()+" set ");

for(String fname:fieldNames){

Object fvalue = ReflectUtils.invokeGet(fname,obj);

params.add(fvalue);

sql.append(fname+"=?,");

}

sql.setCharAt(sql.length()-1, ' ');

sql.append(" where ");

sql.append(priKey.getName()+"=? ");

params.add(ReflectUtils.invokeGet(priKey.getName(), obj));    //主键的值

return executeDML(sql.toString(), params.toArray());

}

}

## reflectUtils类：

package com.bjsxt.sorm.utils;

import java.lang.reflect.InvocationTargetException;

import java.lang.reflect.Method;

/\*\*

 \* 封装了反射常用的操作

 \* @author gaoqi www.sxt.cn

 \*

 \*/

public class ReflectUtils {

/\*\*

 \* 调用obj对象对应属性fieldName的get方法

 \* @param fieldName

 \* @param obj

 \* @return

 \*/

public static Object invokeGet(String fieldName,Object obj){

try {

Class c = obj.getClass();

Method m = c.getDeclaredMethod("get"+StringUtils.firstChar2UpperCase(fieldName), null);

return m.invoke(obj, null);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

public static void invokeSet(Object obj,String columnName,Object columnValue){

try {

Method m = obj.getClass().getDeclaredMethod("set"+StringUtils.firstChar2UpperCase(columnName),

columnValue.getClass());

m.invoke(obj, columnValue);

} catch (Exception e) {

e.printStackTrace();

}

}

}

# 第289-290集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季289--290集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# ＳＯＲＭ进阶:

    本集主要是对实现将一个对象直接存储到数据里面的一个操作:

    MysqlQuery里面Insert方法，update方法的操作

    289--Insert();

    290--update();

package com.bjsxt.sorm.core;

import java.lang.reflect.Field;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.util.ArrayList;

import java.util.List;

import com.bjsxt.po.Emp;

import com.bjsxt.sorm.bean.ColumnInfo;

import com.bjsxt.sorm.bean.TableInfo;

import com.bjsxt.sorm.utils.JDBCUtils;

import com.bjsxt.sorm.utils.ReflectUtils;

import com.bjsxt.vo.EmpVO;

/\*\*

 \* 负责针对Mysql数据库的查询

 \* @author gaoqi

 \*

 \*/

public class MySqlQuery implements Query {

public static void testDML(){

Emp e = new Emp();

e.setEmpname("lily");

e.setBirthday(new java.sql.Date(System.currentTimeMillis()));

e.setAge(30);

e.setSalary(3000.8);

e.setId(1);

// new MySqlQuery().delete(e);

// new MySqlQuery().insert(e);

new MySqlQuery().update(e,new String[]{"empname","age","salary"});

}

public static void testQueryRows(){

List<Emp> list = new MySqlQuery().queryRows("select id,empname,age from emp where age>? and salary<?",

Emp.class, new Object[]{10,5000});

for(Emp e:list){

System.out.println(e.getEmpname());

}

String sql2 = "select e.id,e.empname,salary+bonus 'xinshui',age,d.dname 'deptName',d.address 'deptAddr' from emp e "

+"join dept d on e.deptId=d.id ";

List<EmpVO> list2 = new MySqlQuery().queryRows(sql2,

EmpVO.class, null);

for(EmpVO e:list2){

System.out.println(e.getEmpname()+"-"+e.getDeptAddr()+"-"+e.getXinshui());

}

}

public static void main(String[] args) {

// Number obj = (Number)new MySqlQuery().queryValue("select count(\*) from emp where salary>?",new Object[]{1000});

Number obj = new MySqlQuery().queryNumber("select count(\*) from emp where salary>?",new Object[]{1000});

System.out.println(obj.doubleValue());

}

@Override

public void delete(Class clazz, Object id) {

//Emp.class,2-->delete from emp where id=2

//通过Class对象找TableInfo

TableInfo tableInfo = TableContext.poClassTableMap.get(clazz);

//获得主键

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();

String sql = "delete from "+tableInfo.getTname()+" where "+onlyPriKey.getName()+"=? ";

executeDML(sql, new Object[]{id});

}

@Override

public void delete(Object obj) {

Class c = obj.getClass();

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();  //主键

//通过反射机制，调用属性对应的get方法或set方法

Object priKeyValue = ReflectUtils.invokeGet(onlyPriKey.getName(), obj);

delete(c, priKeyValue);

}

@Override

public int executeDML(String sql, Object[] params) {

Connection conn = DBManager.getConn();

int count = 0;

PreparedStatement ps = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

count  = ps.executeUpdate();

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return count;

}

@Override

public void insert(Object obj) {

//obj-->表中。             insert into 表名  (id,uname,pwd) values (?,?,?)

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

StringBuilder sql  = new StringBuilder("insert into "+tableInfo.getTname()+" (");

int countNotNullField = 0;   //计算不为null的属性值

Field[] fs = c.getDeclaredFields();

for(Field f:fs){

String fieldName = f.getName();

Object fieldValue = ReflectUtils.invokeGet(fieldName, obj);

if(fieldValue!=null){

countNotNullField++;

sql.append(fieldName+",");

params.add(fieldValue);

}

}

sql.setCharAt(sql.length()-1, ')');

sql.append(" values (");

for(int i=0;i<countNotNullField;i++){

sql.append("?,");

}

sql.setCharAt(sql.length()-1, ')');

executeDML(sql.toString(), params.toArray());

}

@Override

public Number queryNumber(String sql, Object[] params) {

return (Number)queryValue(sql, params);

}

@Override

public List queryRows(String sql, Class clazz, Object[] params) {

Connection conn = DBManager.getConn();

List list = null;    //存储查询结果的容器

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

ResultSetMetaData metaData = rs.getMetaData();

//多行

while(rs.next()){

if(list==null){

list = new ArrayList();

}

Object rowObj = clazz.newInstance();   //调用javabean的无参构造器

//多列       select username ,pwd,age from user where id>? and age>18

for(int i=0;i<metaData.getColumnCount();i++){

String columnName = metaData.getColumnLabel(i+1);  //username

Object columnValue = rs.getObject(i+1);

//调用rowObj对象的setUsername(String uname)方法，将columnValue的值设置进去

ReflectUtils.invokeSet(rowObj, columnName, columnValue);

}

list.add(rowObj);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return list;

}

@Override

public Object queryUniqueRow(String sql, Class clazz, Object[] params) {

List list = queryRows(sql, clazz, params);

return (list==null&&list.size()>0)?null:list.get(0);

}

@Override

public Object queryValue(String sql, Object[] params) {

Connection conn = DBManager.getConn();

Object value = null;    //存储查询结果的对象

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

while(rs.next()){

value = rs.getObject(1);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return value;

}

@Override

public int update(Object obj, String[] fieldNames) {

//obj{"uanme","pwd"}-->update 表名  set uname=?,pwd=? where id=?

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo  priKey = tableInfo.getOnlyPriKey();   //获得唯一的主键

StringBuilder sql  = new StringBuilder("update "+tableInfo.getTname()+" set ");

for(String fname:fieldNames){

Object fvalue = ReflectUtils.invokeGet(fname,obj);

params.add(fvalue);

sql.append(fname+"=?,");

}

sql.setCharAt(sql.length()-1, ' ');

sql.append(" where ");

sql.append(priKey.getName()+"=? ");

params.add(ReflectUtils.invokeGet(priKey.getName(), obj));    //主键的值

return executeDML(sql.toString(), params.toArray());

}

}

# 第291-292集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季291--292集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM进阶：

    291--queryRows();

    292--EmpVo(值对象的添加)：针对联合查询。

    sql2 多表查询的测试

## mysqlQuery当中的querRows方法的封装：

package com.bjsxt.sorm.core;

import java.lang.reflect.Field;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.util.ArrayList;

import java.util.List;

import com.bjsxt.po.Emp;

import com.bjsxt.sorm.bean.ColumnInfo;

import com.bjsxt.sorm.bean.TableInfo;

import com.bjsxt.sorm.utils.JDBCUtils;

import com.bjsxt.sorm.utils.ReflectUtils;

import com.bjsxt.vo.EmpVO;

/\*\*

 \* 负责针对Mysql数据库的查询

 \* @author gaoqi

 \*

 \*/

public class MySqlQuery implements Query {

public static void testDML(){

Emp e = new Emp();

e.setEmpname("lily");

e.setBirthday(new java.sql.Date(System.currentTimeMillis()));

e.setAge(30);

e.setSalary(3000.8);

e.setId(1);

// new MySqlQuery().delete(e);

// new MySqlQuery().insert(e);

new MySqlQuery().update(e,new String[]{"empname","age","salary"});

}

public static void testQueryRows(){

List<Emp> list = new MySqlQuery().queryRows("select id,empname,age from emp where age>? and salary<?",

Emp.class, new Object[]{10,5000});

for(Emp e:list){

System.out.println(e.getEmpname());

}

String sql2 = "select e.id,e.empname,salary+bonus 'xinshui',age,d.dname 'deptName',d.address 'deptAddr' from emp e "

+"join dept d on e.deptId=d.id ";

List<EmpVO> list2 = new MySqlQuery().queryRows(sql2,

EmpVO.class, null);

for(EmpVO e:list2){

System.out.println(e.getEmpname()+"-"+e.getDeptAddr()+"-"+e.getXinshui());

}

}

public static void main(String[] args) {

// Number obj = (Number)new MySqlQuery().queryValue("select count(\*) from emp where salary>?",new Object[]{1000});

Number obj = new MySqlQuery().queryNumber("select count(\*) from emp where salary>?",new Object[]{1000});

System.out.println(obj.doubleValue());

}

@Override

public void delete(Class clazz, Object id) {

//Emp.class,2-->delete from emp where id=2

//通过Class对象找TableInfo

TableInfo tableInfo = TableContext.poClassTableMap.get(clazz);

//获得主键

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();

String sql = "delete from "+tableInfo.getTname()+" where "+onlyPriKey.getName()+"=? ";

executeDML(sql, new Object[]{id});

}

@Override

public void delete(Object obj) {

Class c = obj.getClass();

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();  //主键

//通过反射机制，调用属性对应的get方法或set方法

Object priKeyValue = ReflectUtils.invokeGet(onlyPriKey.getName(), obj);

delete(c, priKeyValue);

}

@Override

public int executeDML(String sql, Object[] params) {

Connection conn = DBManager.getConn();

int count = 0;

PreparedStatement ps = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

count  = ps.executeUpdate();

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return count;

}

@Override

public void insert(Object obj) {

//obj-->表中。             insert into 表名  (id,uname,pwd) values (?,?,?)

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

StringBuilder sql  = new StringBuilder("insert into "+tableInfo.getTname()+" (");

int countNotNullField = 0;   //计算不为null的属性值

Field[] fs = c.getDeclaredFields();

for(Field f:fs){

String fieldName = f.getName();

Object fieldValue = ReflectUtils.invokeGet(fieldName, obj);

if(fieldValue!=null){

countNotNullField++;

sql.append(fieldName+",");

params.add(fieldValue);

}

}

sql.setCharAt(sql.length()-1, ')');

sql.append(" values (");

for(int i=0;i<countNotNullField;i++){

sql.append("?,");

}

sql.setCharAt(sql.length()-1, ')');

executeDML(sql.toString(), params.toArray());

}

@Override

public Number queryNumber(String sql, Object[] params) {

return (Number)queryValue(sql, params);

}

@Override

public List queryRows(String sql, Class clazz, Object[] params) {

Connection conn = DBManager.getConn();

List list = null;    //存储查询结果的容器

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

ResultSetMetaData metaData = rs.getMetaData();

//多行

while(rs.next()){

if(list==null){

list = new ArrayList();

}

Object rowObj = clazz.newInstance();   //调用javabean的无参构造器

//多列       select username ,pwd,age from user where id>? and age>18

for(int i=0;i<metaData.getColumnCount();i++){

String columnName = metaData.getColumnLabel(i+1);  //username

Object columnValue = rs.getObject(i+1);

//调用rowObj对象的setUsername(String uname)方法，将columnValue的值设置进去

ReflectUtils.invokeSet(rowObj, columnName, columnValue);

}

list.add(rowObj);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return list;

}

@Override

public Object queryUniqueRow(String sql, Class clazz, Object[] params) {

List list = queryRows(sql, clazz, params);

return (list==null&&list.size()>0)?null:list.get(0);

}

@Override

public Object queryValue(String sql, Object[] params) {

Connection conn = DBManager.getConn();

Object value = null;    //存储查询结果的对象

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

while(rs.next()){

value = rs.getObject(1);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return value;

}

@Override

public int update(Object obj, String[] fieldNames) {

//obj{"uanme","pwd"}-->update 表名  set uname=?,pwd=? where id=?

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo  priKey = tableInfo.getOnlyPriKey();   //获得唯一的主键

StringBuilder sql  = new StringBuilder("update "+tableInfo.getTname()+" set ");

for(String fname:fieldNames){

Object fvalue = ReflectUtils.invokeGet(fname,obj);

params.add(fvalue);

sql.append(fname+"=?,");

}

sql.setCharAt(sql.length()-1, ' ');

sql.append(" where ");

sql.append(priKey.getName()+"=? ");

params.add(ReflectUtils.invokeGet(priKey.getName(), obj));    //主键的值

return executeDML(sql.toString(), params.toArray());

}

}

## EmpVo对象的封装：这个对象是我们根据实际情况查询的出的结果而封装的。

package com.bjsxt.vo;

public class EmpVO {

private Integer id;

private String empname;

private Double xinshui;

private Integer age;

private String deptName;

private String deptAddr;

public EmpVO(Integer id, String empname, Double xinshui, Integer age,

String deptName, String deptAddr) {

super();

this.id = id;

this.empname = empname;

this.xinshui = xinshui;

this.age = age;

this.deptName = deptName;

this.deptAddr = deptAddr;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getEmpname() {

return empname;

}

public void setEmpname(String empname) {

this.empname = empname;

}

public Double getXinshui() {

return xinshui;

}

public void setXinshui(Double xinshui) {

this.xinshui = xinshui;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

public String getDeptName() {

return deptName;

}

public void setDeptName(String deptName) {

this.deptName = deptName;

}

public String getDeptAddr() {

return deptAddr;

}

public void setDeptAddr(String deptAddr) {

this.deptAddr = deptAddr;

}

public EmpVO() {

}

}

# 第293集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季293集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM进阶：

   293-- mysqlQuery当中    queryValue方法的封装：(当行单列)测试

                                     queryNumber();测试：

    280--293版本终结，以后是在这个版本的基础上进行修复了！~

mysqlQuery--

package com.bjsxt.sorm.core;

import java.lang.reflect.Field;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.util.ArrayList;

import java.util.List;

import com.bjsxt.po.Emp;

import com.bjsxt.sorm.bean.ColumnInfo;

import com.bjsxt.sorm.bean.TableInfo;

import com.bjsxt.sorm.utils.JDBCUtils;

import com.bjsxt.sorm.utils.ReflectUtils;

import com.bjsxt.vo.EmpVO;

/\*\*

 \* 负责针对Mysql数据库的查询

 \* @author gaoqi

 \*

 \*/

public class MySqlQuery implements Query {

public static void testDML(){

Emp e = new Emp();

e.setEmpname("lily");

e.setBirthday(new java.sql.Date(System.currentTimeMillis()));

e.setAge(30);

e.setSalary(3000.8);

e.setId(1);

// new MySqlQuery().delete(e);

// new MySqlQuery().insert(e);

new MySqlQuery().update(e,new String[]{"empname","age","salary"});

}

public static void testQueryRows(){

List<Emp> list = new MySqlQuery().queryRows("select id,empname,age from emp where age>? and salary<?",

Emp.class, new Object[]{10,5000});

for(Emp e:list){

System.out.println(e.getEmpname());

}

String sql2 = "select e.id,e.empname,salary+bonus 'xinshui',age,d.dname 'deptName',d.address 'deptAddr' from emp e "

+"join dept d on e.deptId=d.id ";

List<EmpVO> list2 = new MySqlQuery().queryRows(sql2,

EmpVO.class, null);

for(EmpVO e:list2){

System.out.println(e.getEmpname()+"-"+e.getDeptAddr()+"-"+e.getXinshui());

}

}

public static void main(String[] args) {

// Number obj = (Number)new MySqlQuery().queryValue("select count(\*) from emp where salary>?",new Object[]{1000});

Number obj = new MySqlQuery().queryNumber("select count(\*) from emp where salary>?",new Object[]{1000});

System.out.println(obj.doubleValue());

}

@Override

public void delete(Class clazz, Object id) {

//Emp.class,2-->delete from emp where id=2

//通过Class对象找TableInfo

TableInfo tableInfo = TableContext.poClassTableMap.get(clazz);

//获得主键

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();

String sql = "delete from "+tableInfo.getTname()+" where "+onlyPriKey.getName()+"=? ";

executeDML(sql, new Object[]{id});

}

@Override

public void delete(Object obj) {

Class c = obj.getClass();

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();  //主键

//通过反射机制，调用属性对应的get方法或set方法

Object priKeyValue = ReflectUtils.invokeGet(onlyPriKey.getName(), obj);

delete(c, priKeyValue);

}

@Override

public int executeDML(String sql, Object[] params) {

Connection conn = DBManager.getConn();

int count = 0;

PreparedStatement ps = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

count  = ps.executeUpdate();

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return count;

}

@Override

public void insert(Object obj) {

//obj-->表中。             insert into 表名  (id,uname,pwd) values (?,?,?)

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

StringBuilder sql  = new StringBuilder("insert into "+tableInfo.getTname()+" (");

int countNotNullField = 0;   //计算不为null的属性值

Field[] fs = c.getDeclaredFields();

for(Field f:fs){

String fieldName = f.getName();

Object fieldValue = ReflectUtils.invokeGet(fieldName, obj);

if(fieldValue!=null){

countNotNullField++;

sql.append(fieldName+",");

params.add(fieldValue);

}

}

sql.setCharAt(sql.length()-1, ')');

sql.append(" values (");

for(int i=0;i<countNotNullField;i++){

sql.append("?,");

}

sql.setCharAt(sql.length()-1, ')');

executeDML(sql.toString(), params.toArray());

}

@Override

public Number queryNumber(String sql, Object[] params) {

return (Number)queryValue(sql, params);

}

@Override

public List queryRows(String sql, Class clazz, Object[] params) {

Connection conn = DBManager.getConn();

List list = null;    //存储查询结果的容器

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

ResultSetMetaData metaData = rs.getMetaData();

//多行

while(rs.next()){

if(list==null){

list = new ArrayList();

}

Object rowObj = clazz.newInstance();   //调用javabean的无参构造器

//多列       select username ,pwd,age from user where id>? and age>18

for(int i=0;i<metaData.getColumnCount();i++){

String columnName = metaData.getColumnLabel(i+1);  //username

Object columnValue = rs.getObject(i+1);

//调用rowObj对象的setUsername(String uname)方法，将columnValue的值设置进去

ReflectUtils.invokeSet(rowObj, columnName, columnValue);

}

list.add(rowObj);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return list;

}

@Override

public Object queryUniqueRow(String sql, Class clazz, Object[] params) {

List list = queryRows(sql, clazz, params);

return (list==null&&list.size()>0)?null:list.get(0);

}

@Override

public Object queryValue(String sql, Object[] params) {

Connection conn = DBManager.getConn();

Object value = null;    //存储查询结果的对象

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

while(rs.next()){

value = rs.getObject(1);

}

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return value;

}

@Override

public int update(Object obj, String[] fieldNames) {

//obj{"uanme","pwd"}-->update 表名  set uname=?,pwd=? where id=?

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo  priKey = tableInfo.getOnlyPriKey();   //获得唯一的主键

StringBuilder sql  = new StringBuilder("update "+tableInfo.getTname()+" set ");

for(String fname:fieldNames){

Object fvalue = ReflectUtils.invokeGet(fname,obj);

params.add(fvalue);

sql.append(fname+"=?,");

}

sql.setCharAt(sql.length()-1, ' ');

sql.append(" where ");

sql.append(priKey.getName()+"=? ");

params.add(ReflectUtils.invokeGet(priKey.getName(), obj));    //主键的值

return executeDML(sql.toString(), params.toArray());

}

# 第294-297集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季294--297集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM代码的优化：

    将MySqlQuery类中的方法放到父类里面去等等策略。

    讲一些方法整合，去除！

## Query.类的进一步优化：

package com.bjsxt.sorm.core;

import java.lang.reflect.Field;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import com.bjsxt.sorm.bean.ColumnInfo;

import com.bjsxt.sorm.bean.TableInfo;

import com.bjsxt.sorm.utils.JDBCUtils;

import com.bjsxt.sorm.utils.ReflectUtils;

/\*\*

 \* 负责查询（对外提供服务的核心类）

 \* @author gaoqi ww.sxt.cn

 \*

 \*/

@SuppressWarnings("all")

public abstract class Query implements Cloneable {

/\*\*

 \* 采用模板方法模式将JDBC操作封装成模板，便于重用

 \* @param sql sql语句

 \* @param params sql的参数

 \* @param clazz 记录要封装到的java类

 \* @param back CallBack的实现类，实现回调

 \* @return

 \*/

public Object executeQueryTemplate(String sql,Object[] params,Class clazz,CallBack back){

Connection conn = DBManager.getConn();

PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

rs = ps.executeQuery();

return  back.doExecute(conn, ps, rs);

} catch (Exception e) {

e.printStackTrace();

return null;

}finally{

DBManager.close(ps, conn);

}

}

/\*\*

 \* 直接执行一个DML语句

 \* @param sql sql语句

 \* @param params 参数

 \* @return 执行sql语句后影响记录的行数

 \*/

public int executeDML(String sql,Object[] params){

Connection conn = DBManager.getConn();

int count = 0;

PreparedStatement ps = null;

try {

ps = conn.prepareStatement(sql);

//给sql设参

JDBCUtils.handleParams(ps, params);

System.out.println(ps);

count  = ps.executeUpdate();

} catch (Exception e) {

e.printStackTrace();

}finally{

DBManager.close(ps, conn);

}

return count;

}

/\*\*

 \* 将一个对象存储到数据库中

 \* 把对象中不为null的属性往数据库中存储！如果数字为null则放0.

 \* @param obj 要存储的对象

 \*/

public void insert(Object obj){

//obj-->表中。             insert into 表名  (id,uname,pwd) values (?,?,?)

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

StringBuilder sql  = new StringBuilder("insert into "+tableInfo.getTname()+" (");

int countNotNullField = 0;   //计算不为null的属性值

Field[] fs = c.getDeclaredFields();

for(Field f:fs){

String fieldName = f.getName();

Object fieldValue = ReflectUtils.invokeGet(fieldName, obj);

if(fieldValue!=null){

countNotNullField++;

sql.append(fieldName+",");

params.add(fieldValue);

}

}

sql.setCharAt(sql.length()-1, ')');

sql.append(" values (");

for(int i=0;i<countNotNullField;i++){

sql.append("?,");

}

sql.setCharAt(sql.length()-1, ')');

executeDML(sql.toString(), params.toArray());

}

/\*\*

 \* 删除clazz表示类对应的表中的记录(指定主键值id的记录)

 \* @param clazz 跟表对应的类的Class对象

 \* @param id 主键的值

 \*/

public void delete(Class clazz,Object id){

//Emp.class,2-->delete from emp where id=2

//通过Class对象找TableInfo

TableInfo tableInfo = TableContext.poClassTableMap.get(clazz);

//获得主键

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();

String sql = "delete from "+tableInfo.getTname()+" where "+onlyPriKey.getName()+"=? ";

executeDML(sql, new Object[]{id});

}

/\*\*

 \* 删除对象在数据库中对应的记录(对象所在的类对应到表，对象的主键的值对应到记录)

 \* @param obj

 \*/

public void delete(Object obj){

Class c = obj.getClass();

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();  //主键

//通过反射机制，调用属性对应的get方法或set方法

Object priKeyValue = ReflectUtils.invokeGet(onlyPriKey.getName(), obj);

delete(c, priKeyValue);

}

/\*\*

 \* 更新对象对应的记录，并且只更新指定的字段的值

 \* @param obj 所要更新的对象

 \* @param fieldNames 更新的属性列表

 \* @return 执行sql语句后影响记录的行数

 \*/

public int update(Object obj,String[] fieldNames){

//obj{"uanme","pwd"}-->update 表名  set uname=?,pwd=? where id=?

Class c = obj.getClass();

List<Object> params = new ArrayList<Object>();   //存储sql的参数对象

TableInfo tableInfo = TableContext.poClassTableMap.get(c);

ColumnInfo  priKey = tableInfo.getOnlyPriKey();   //获得唯一的主键

StringBuilder sql  = new StringBuilder("update "+tableInfo.getTname()+" set ");

for(String fname:fieldNames){

Object fvalue = ReflectUtils.invokeGet(fname,obj);

params.add(fvalue);

sql.append(fname+"=?,");

}

sql.setCharAt(sql.length()-1, ' ');

sql.append(" where ");

sql.append(priKey.getName()+"=? ");

params.add(ReflectUtils.invokeGet(priKey.getName(), obj));    //主键的值

return executeDML(sql.toString(), params.toArray());

}

/\*\*

 \* 查询返回多行记录，并将每行记录封装到clazz指定的类的对象中

 \* @param sql 查询语句

 \* @param clazz 封装数据的javabean类的Class对象

 \* @param params sql的参数

 \* @return 查询到的结果

 \*/

public List queryRows(final String sql,final Class clazz,final Object[] params){

return (List)executeQueryTemplate(sql, params, clazz, new CallBack() {

@Override

public Object doExecute(Connection conn, PreparedStatement ps, ResultSet rs) {

List list = null;

try {

ResultSetMetaData metaData = rs.getMetaData();

//多行

while(rs.next()){

if(list==null){

list = new ArrayList();

}

Object rowObj = clazz.newInstance();   //调用javabean的无参构造器

//多列       select username ,pwd,age from user where id>? and age>18

for(int i=0;i<metaData.getColumnCount();i++){

String columnName = metaData.getColumnLabel(i+1);  //username

Object columnValue = rs.getObject(i+1);

//调用rowObj对象的setUsername(String uname)方法，将columnValue的值设置进去

ReflectUtils.invokeSet(rowObj, columnName, columnValue);

}

list.add(rowObj);

}

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

});

}

/\*\*

 \* 查询返回一行记录，并将该记录封装到clazz指定的类的对象中

 \* @param sql 查询语句

 \* @param clazz 封装数据的javabean类的Class对象

 \* @param params sql的参数

 \* @return 查询到的结果

 \*/

public Object queryUniqueRow(String sql,Class clazz,Object[] params){

List list = queryRows(sql, clazz, params);

return (list!=null&&list.size()>0)?list.get(0):null;

}

/\*\*

 \* 根据主键的值直接查找对应的对象

 \* @param clazz

 \* @param id

 \* @return

 \*/

public Object queryById(Class clazz,Object id){

//select \* from emp where id=?   //delete from emp where id=?

TableInfo tableInfo = TableContext.poClassTableMap.get(clazz);

//获得主键

ColumnInfo onlyPriKey = tableInfo.getOnlyPriKey();

String sql = "select \* from "+tableInfo.getTname()+" where "+onlyPriKey.getName()+"=? ";

return queryUniqueRow(sql, clazz, new Object[]{id});

}

/\*\*

 \* 查询返回一个值(一行一列)，并将该值返回

 \* @param sql 查询语句

 \* @param params sql的参数

 \* @return 查询到的结果

 \*/

public Object queryValue(String sql,Object[] params){

return executeQueryTemplate(sql, params, null, new CallBack() {

@Override

public Object doExecute(Connection conn, PreparedStatement ps, ResultSet rs) {

Object value = null;

try {

while(rs.next()){

value = rs.getObject(1);

}

} catch (SQLException e) {

e.printStackTrace();

}

return value;

}

});

}

/\*\*

 \* 查询返回一个数字(一行一列)，并将该值返回

 \* @param sql 查询语句

 \* @param params sql的参数

 \* @return 查询到的数字

 \*/

public Number queryNumber(String sql,Object[] params){

return (Number)queryValue(sql, params);

}

/\*\*

 \* 分页查询

 \* @param pageNum 第几页数据

 \* @param size 每页显示多少记录

 \* @return

 \*/

public abstract Object queryPagenate(int pageNum,int size);

@Override

protected Object clone() throws CloneNotSupportedException {

return super.clone();

}

}

## CallBack类封装

package com.bjsxt.sorm.core;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public interface CallBack {

public Object doExecute(Connection conn,PreparedStatement ps,ResultSet rs);

}

## MySqlQuery的进一步优化：

295集--创建Query对象的工厂类QueryFactory，

    工厂类的单列模式

package com.bjsxt.sorm.core;

/\*\*

 \* 创建Query对象的工厂类

 \* @author gaoqi

 \*

 \*/

public class QueryFactory {

private static Query prototypeObj;  //原型对象

static {

try {

Class c = Class.forName(DBManager.getConf().getQueryClass());  ////加载指定的query类

prototypeObj = (Query) c.newInstance();

} catch (Exception e) {

e.printStackTrace();

}

//加载po包下面所有的类，便于重用，提高效率！

TableContext.loadPOTables();

}

private QueryFactory(){  //私有构造器

}

public static Query createQuery(){

try {

return (Query) prototypeObj.clone();

} catch (CloneNotSupportedException e) {

e.printStackTrace();

return null;

}

}

}

296--增加连接池优化sorm

        DBConnPool

        DBManager.java

        Configiguration

package com.bjsxt.sorm.pool;

import java.lang.reflect.InvocationHandler;

import java.lang.reflect.Proxy;

import java.sql.Connection;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import com.bjsxt.sorm.core.DBManager;

/\*\*

 \* 连接池的类

 \* @author gaoqi

 \*

 \*/

public class DBConnPool {

/\*\*

 \* 连接池对象

 \*/

private  List<Connection> pool;

/\*\*

 \* 最大连接数

 \*/

private static final int POOL\_MAX\_SIZE = DBManager.getConf().getPoolMaxSize();

/\*\*

 \* 最小连接池

 \*/

private static final int POOL\_MIN\_SIZE = DBManager.getConf().getPoolMinSize();

/\*\*

 \* 初始化连接池，使池中的连接数达到最小值

 \*/

public void initPool() {

if(pool==null){

pool = new ArrayList<Connection>();

}

while(pool.size()<DBConnPool.POOL\_MIN\_SIZE){

pool.add(DBManager.createConn());

System.out.println("初始化池，池中连接数："+pool.size());

}

}

/\*\*

 \* 从连接池中取出一个连接

 \* @return

 \*/

public synchronized Connection getConnection() {

int last\_index = pool.size()-1;

Connection conn = pool.get(last\_index);

pool.remove(last\_index);

return conn;

}

/\*\*

 \* 将连接放回池中

 \* @param conn

 \*/

public synchronized void close(Connection conn){

if(pool.size()>=POOL\_MAX\_SIZE){

try {

if(conn!=null){

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}else {

pool.add(conn);

}

}

public DBConnPool() {

initPool();

}

}

# 第298集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季298集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM项目的整理导出：

    1）整理导出项目

    2）生成相应的API文档：

    3）初学者如何把项目生成相应的api文档，观看视频就可以了。

    4）整理后的项目包含了sorm0.8.jar,API文档，rendme.txt等等。

# 本集视频没有代码可以整理，项目的尽量多参考视频即可，可以到三百集里寻找源码参考！

# 第299集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季299集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM的测试：

    1）配置文件的修改

    2）功能的测试：

    3）测试的准备工作，首先要把所用的数据库的驱动夹包，和项目的架包引入都我们的醒目中。

driver=com.mysql.jdbc.Driver

url=jdbc\:mysql\://localhost\:3306/sorm

user=root

pwd=123456

usingDB=mysql

srcPath=D\:\\workspace\\SORM\_test001\\src

poPackage=com.bjsxt.test.po

queryClass=com.bjsxt.sorm.core.MySqlQuery

poolMinSize=10

poolMaxSize=100

## 测试的代码如下：

### po类代码

package com.bjsxt.test.po;

import java.sql.\*;

import java.util.\*;

public class Emp {

private Integer id;

private java.sql.Date birthday;

private Integer age;

private String empname;

private Double bonus;

private Double salary;

private Integer deptId;

public Integer getId(){

return id;

}

public java.sql.Date getBirthday(){

return birthday;

}

public Integer getAge(){

return age;

}

public String getEmpname(){

return empname;

}

public Double getBonus(){

return bonus;

}

public Double getSalary(){

return salary;

}

public Integer getDeptId(){

return deptId;

}

public void setId(Integer id){

this.id=id;

}

public void setBirthday(java.sql.Date birthday){

this.birthday=birthday;

}

public void setAge(Integer age){

this.age=age;

}

public void setEmpname(String empname){

this.empname=empname;

}

public void setBonus(Double bonus){

this.bonus=bonus;

}

public void setSalary(Double salary){

this.salary=salary;

}

public void setDeptId(Integer deptId){

this.deptId=deptId;

}

}

package com.bjsxt.test.po;

import java.sql.\*;

import java.util.\*;

public class Dept {

private Integer id;

private String address;

private String dname;

public Integer getId(){

return id;

}

public String getAddress(){

return address;

}

public String getDname(){

return dname;

}

public void setId(Integer id){

this.id=id;

}

public void setAddress(String address){

this.address=address;

}

public void setDname(String dname){

this.dname=dname;

}

}

## 测试功能的代码如下：

package test;

import java.util.List;

import com.bjsxt.sorm.core.Query;

import com.bjsxt.sorm.core.QueryFactory;

import com.bjsxt.test.po.Emp;

import com.bjsxt.test.vo.EmpVO;

public class Test {

public static void add(){

Emp e = new Emp();

e.setAge(18);

e.setEmpname("高琪琪");

e.setSalary(2000.0);

Query q  = QueryFactory.createQuery();

q.insert(e);

}

public static void delete(){

Emp e = new Emp();

e.setAge(18);

e.setEmpname("高琪琪");

e.setSalary(2000.0);

e.setId(3);

Query q  = QueryFactory.createQuery();

q.delete(e);

}

public static void update(){

Emp e = new Emp();

e.setAge(180);

e.setEmpname("高琪琪");

e.setSalary(200.0);

e.setId(1);

Query q  = QueryFactory.createQuery();

q.update(e, new String[]{"age","salary"});

}

public static void select01(){

Query q  = QueryFactory.createQuery();

Number n = q.queryNumber("select count(\*) from emp where salary>?",new Object[]{100});

System.out.println(n);

}

public static void select02(){

Query q  = QueryFactory.createQuery();

Emp e =  (Emp)q.queryUniqueRow("select \* from emp where id=?", Emp.class, new Object[]{40});

System.out.println(e);

System.out.println(e.getEmpname());

}

public static void select03(){

Query q  = QueryFactory.createQuery();

List<Emp> list =  (List)q.queryRows("select \* from emp where id>?", Emp.class, new Object[]{41});

System.out.println(list);

for (Emp emp : list) {

System.out.println(emp.getEmpname()+"--"+emp.getBirthday());

}

}

public static void select04(){

Query q  = QueryFactory.createQuery();

String sql = "select e.id,e.empname,e.age,d.dname 'deptName',d.address 'deptAddr' from emp e join dept d on e.deptId=d.id ";

List<EmpVO> list =  (List)q.queryRows(sql,

EmpVO.class, new Object[]{});

System.out.println(list);

for (EmpVO emp : list) {

System.out.println(emp.getEmpname()+"--"+emp.getDeptAddr()+"--"+emp.getDeptName());

}

}

public static void test05(){

Query q  = QueryFactory.createQuery();

Emp e = (Emp) q.queryById(Emp.class, 43);

System.out.println(e.getEmpname());

}

public static void main(String[] args) {

//通过这个方法可以生成po类！

// TableContext.updateJavaPOFile();

// select01();

// delete();

// update();

test05();

}

}

# 第300集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季300集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# SORM的测试的增加

    1）视频299集测试的时候就测试了两个方法，最后这集增加了两个方法，就构成了增删改查的测试，我上集的代码已经发布了最终的测试代码了。这集就不在发布了。

    2）关于这个手写的框架的项目，希望大家多多参考我们的视频，代码也有源码提供。可以下载完整版的项目直接部署到eclipse里就可以了。方便参考。

    3）本集新增的代码，和bug都已经修复了！下载我们视频的地址告诉大家Http://www.bjsxt.com

    我们的视频里选择300集即可！