

# Jxva Framework 开源框架



作 者：蒋 赞 （Jxvaer）

文档版本：Version 1.0

电子邮箱： [hi@jxva.com](mailto:hi@jxva.com)

官方网站： [www.jxva.com](http://www.jxva.com) [www.jxva.net](http://www.jxva.net) [www.jxva.org](http://www.jxva.org)

## 内容简介

本书的编写主要是为了介绍 Jxva Framework 开源框架快速使用的指导书。围绕 Jxva Framework 基于热插拔核心框架 MVC、基于泛型与注解的 DAO 及数据模型自动生成工具、Tag 基础标签、基于角色的访问控制体系 RBAC、OXM 对象-XML 映射、通用的跨域单点登录 SSO、高性能的 Cache 体系、License 授权认证、基于 Pull 的 XML 文档解析、通用的 JMS 框架、图形切割及缩放、报表生成、文件上传与下载、及通用的基于无限级树型的 JS 解决方案、JS 模拟层框架及下拉菜单等框架的介绍，配合实战范例演示，目的是让使用者和其它感兴趣的开发人员，以最简单直接的方法，快速的进入 JEE 开发。本书所讲解范例都是实际中的真实应用，通过范例的学习，可以加深对所学内容的理解，明白实际项目的开发步骤，最后可以将学到的技巧应用到自己的开发项目中。

本书作者完全展示长达六年的 JEE 独家开发经验，针对 JEE 开发人员在设计中经常遇到的问题及操作技巧进行了全面的剖析，适合于有一定 JAVA 基础的开发人员作为进阶宝典，同时也适合于对 JEE 开源框架感兴趣的朋友，希望从 Jxva Framework 框架的学习中更进一步加深对目前主流开源框架的深入理解。

## 前 言

除了我的家人，软件语言是我的挚爱。没有比软件开发更令我心情舒畅的事情，从我步入编程的日子以来，几乎空闲中 95% 的时间都被给其占用，而在我的生命中最吝啬的就是时间，为了生活我们必须不断学习来维持在社会工作中的提升与进取。

我们不是机器，我们也不是万能的，在有限的时间里，在无限的开源世界里，我们面对的开源框架太多了 Struts、Hibernate、Spring、WebWork...，没有这么多精力与时间去不断的学习，在此我向您推荐 Jxva Framework 框架，Jxva Framework 是一个开放源代码的、基于热插拔功能扩展的、超轻量级的、快速开发的、不依赖任何第三方框架的 JEE 框架。

或许您正在学习 Java 技术，为漫天的 Java 新名词而苦恼，如 Design Patterns、Refactoring、Agile、Xp、RUP、Struts、Hibernate、Spring、WebWork、EJB、VO、PO、BO、POJO、Generic、Reflect、Annotation 等，从现在开始您可以大胆地放弃这些摸不着头脑而又日新月异的新概念、新框架而无所适存，因为您可以从 Jxva Framework 开源框架使用中深深地学习、学会这些新名词与新概念，Jxva Framework 框架中完全包括了这些新名词与新技术，从对 Jxva Framework 框架的学习你可以深入的理解 Struts、Hibernate、Spring 等目前主流开源框架的内核原理及工作机制。为你的开发工作与技术提升带来质的飞跃。

笔者知识有限，但已经尽力做到最好，本书如有错漏，烦请读者赐教。本书相关知识请浏览本书网页 <http://www.jxva.com/>，有更新文稿、修正源文件、源码范例、业界消息等。

最后我需要感谢我的家人对我的支持与理解，我母亲对我不断追求事业无所成的无怨无悔，感谢在我写此框架与书的过程中一直陪伴在我身边，给了我很大帮助的人，那就是我的女友 ZhuEr。我把最诚挚的感谢献给她。任何珍宝都没有她的陪伴重要。

我希望能喜爱这本书，我希望能因使用 Jxva Framework 开源框架而骄傲，并享受其中的快乐，如果你从本书中略微看到了这种快乐，如果本框架使你

开始感受到了这种骄傲，如果本框架点燃了你内心欣赏这种美的火花，那么就远超过我的目标了。

# 目 录

Jxva Framework 开源框架 .....	I
内容简介 .....	II
前 言 .....	III
目 录 .....	V
1、Jxva Framework 概述 .....	1
1.1 体系结构 .....	1
1.2 框架源码 .....	2
1.2.1 直接下载 Release 版本 .....	2
1.2.2 通过 SVN 同步下载最新框架版本 .....	2
1.2.3 框架原码包名说明 .....	2
1.3 框架插件 .....	3
1.3.1 直接下载框架插件 Release 版本 .....	3
1.3.2 通过 SVN 同步下载最新框架插件版本 .....	3
1.3.3 插件的安装与使用 .....	3
2、利用 Jxva Tool 快速自动生成数据表模型 .....	4
2.1 下载直接运行 Jxva Tool 工具 .....	4
2.2 从源代码工程中运行 .....	5
2.3 利用 JxvaPlugin 插件生成 .....	5
2.4 自动生成工具的使用 .....	6
2.4.1 建立数据表 .....	6
2.4.2 生成数据模型 .....	7
2.5 生成文件说明 .....	9
2.5.1 DBConf.properties 说明 .....	9
2.5.2 数据模型文件说明 .....	10
2.5.3 Hibernate 相关 xml 文档 .....	12
3、DAO 框架 .....	14
3.1 DAO 整体架构及数据支持 .....	14
3.2 创建 DAOFactory .....	14
3.3 创建 DAO .....	15
3.4 关闭 DAO .....	15
3.5 DAO 的操作 .....	16
3.5.1 插入操作 .....	16
3.5.2 更新操作 .....	17
3.5.3 删除操作 .....	17
3.5.4 单个查询 .....	18
3.5.5 单表查询 .....	18
3.5.6 多表查询 .....	20
3.5.7 事务操作 .....	20
3.5.8 使用 SQL .....	20
3.5.9 其它操作 .....	21

3.6 与 Hibernate 的比较.....	21
4、框架核心配置.....	22
5、MVC 框架 .....	23
6、Tag 标签.....	24
7、RBAC 体系.....	25

# 1、Jxva Framework 概述

Jxva Framework 是 Jxvaer 独立研发的一个开放源代码的、基于热插拨功能扩展的、超轻量级的、快速开发的、不依赖任何第三方框架的 JEE 框架及企业资源整合集成平台。利用 Web 方式极大的方便了用户对于系统功能的使用；通过合理的框架组织及泛型与注解的采用，降低了使用者在开发工作中对于快速开发的技术要求。作为国内领先的 JEE 开源框架及资源整合集成平台，Jxva Framework 在研发阶段就瞄准了国际市场，成功借助中软及中兴的成熟技术，集合众多的项目实施经验，使得产品既具有良好的国际通用性，又符合国内的业务特点。

## 1.1 体系结构

### 1) 采用 Java 技术

系统基于 JEE 标准，不依赖于任何第三方开源框架，支持主流的应用服务器平台。

### 2) 支持通用数据库平台 DAO

基于泛型与注解的的 ORM 解决方案，DAO 框架支持通用数据库系统产品,包括 Oracle,DB2,SqlServer,MySql 及嵌入式数据库。

### 3) 支持通用的 MVC 体系

MVC 体系形于 Struts2，而性能及快捷又远超 Struts2，基于注解的 Action 映射减少 XML 文档配置的复杂性

### 4) 完善的基于 RBAC 的用户权限管理体系

基于角色的访问控制 RBAC 体系，系统提供统一的应用权限和用户角色管理，确保授权的人以授权的方式访问授权的信息。

### 5) 门户应用集成

多种类型信息的集成，包括 Web 站点、服务资源以及各种基于 Web 的应用系统。

### 6) 通用的 JMS 消息服务体系

完成支持 IMB MQ、JBossMQ、OpenJMS、ActiveMQ、SUN IMQ

## 7) 通用的跨域 SSO 单点登陆解决方案

解决了集成应用系统之间的一次性统一登陆问题。

## 1.2 框架源码

Jxva Framework框架JAVA工程源码可以从 <http://code.google.com/p/jxva/> 处获得:

### 1.2.1 直接下载 Release 版本

<http://code.google.com/p/jxva/downloads/>

### 1.2.2 通过 SVN 同步下载最新框架版本

<http://code.google.com/p/jxva/downloads/>

### 1.2.3 框架原码包名说明

Eclipse 下的 Jxva Framework 框架 JAVA 工程源码截图如下:

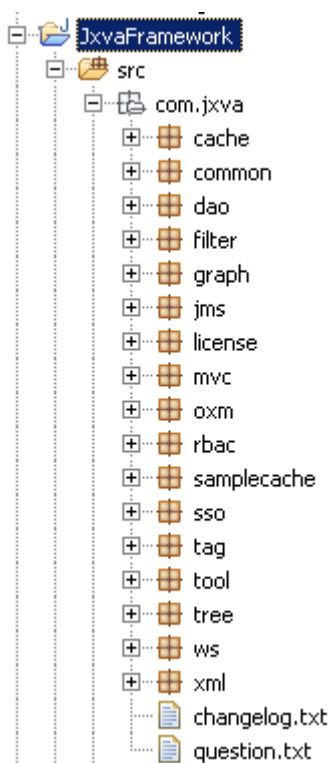


图 1

#### 各包名说明

<b>cache:</b>	缓存框架（基于 ehcache 内核）
<b>common:</b>	公共及实用类
<b>dao:</b>	基于泛型与注解的 ORM 框架
<b>filter:</b>	过滤器
<b>graph:</b>	图形切割及缩放处理框架
<b>jms:</b>	通用的消息服务处理框架
<b>license:</b>	License 授权处理框架
<b>mvc:</b>	MVC 核心框架
<b>oxm:</b>	Object-XML Mapping 处理框架
<b>rbac:</b>	基于角色的访问控制体系
<b>samplecache:</b>	简单的缓存框架
<b>sso:</b>	跨域单点登录体系
<b>tag:</b>	实用标签
<b>tool:</b>	代码辅助生成工具
<b>tree:</b>	无限级树型解决体系
<b>ws:</b>	webservices 实现体系
<b>xml:</b>	基于 Pull 的 XML 解析框架



## 1.3 框架插件

目前基于 Eclipse 的 Jxva Framework 框架插件(JxvaPlugin)的也已着手开发，进展迅速，相信过不了多久将发布，JxvaPlugin 是为了进一步减化开发人员使用 Jxva Framework 框架的工作量，可以自动生成 XML 核心配置文档及 Jxva Framework 工程结构，更加快速与便捷的让开发人员进行开发，关于 JxvaPlugin 的安装与使用将于 7 月份进一步在此文档中完善。

### 1.3.1 直接下载框架插件 Release 版本

<http://code.google.com/p/jxva/downloads/>

### 1.3.2 通过 SVN 同步下载最新框架插件版本

<http://code.google.com/p/jxva/downloads/>

### 1.3.3 插件的安装与使用

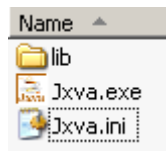
<http://code.google.com/p/jxva/downloads/>

## 2、利用 Jxva Tool 快速自动生成数据表模型

Jxva Tool 是一个代码自动生成工具，目前主要支持数据表模型的生成，同时也支持 hibernate 的 hbm 自动生成，Jxva Tool 对应的源代码位于 Jxva Framework 框架中的 com.jxva.tool 包，支持 Oracle、DB2、Mysql、SqlServer 四种数据库，使用 Jxva Tool 代码自动生成工具，可以有如下两种方法：

### 2.1 下载直接运行 Jxva Tool 工具

从<http://code.google.com/p/jxva/downloads/list> 中可以直接下载JxvaTool.rar工



具，解压到本地如图：

图 2，在确保你已经安装了JRE5 或JDK5 及以上版本环境后，双击Jxva.exe即可运行，运行界面如下图(红色标记是笔者所加)：

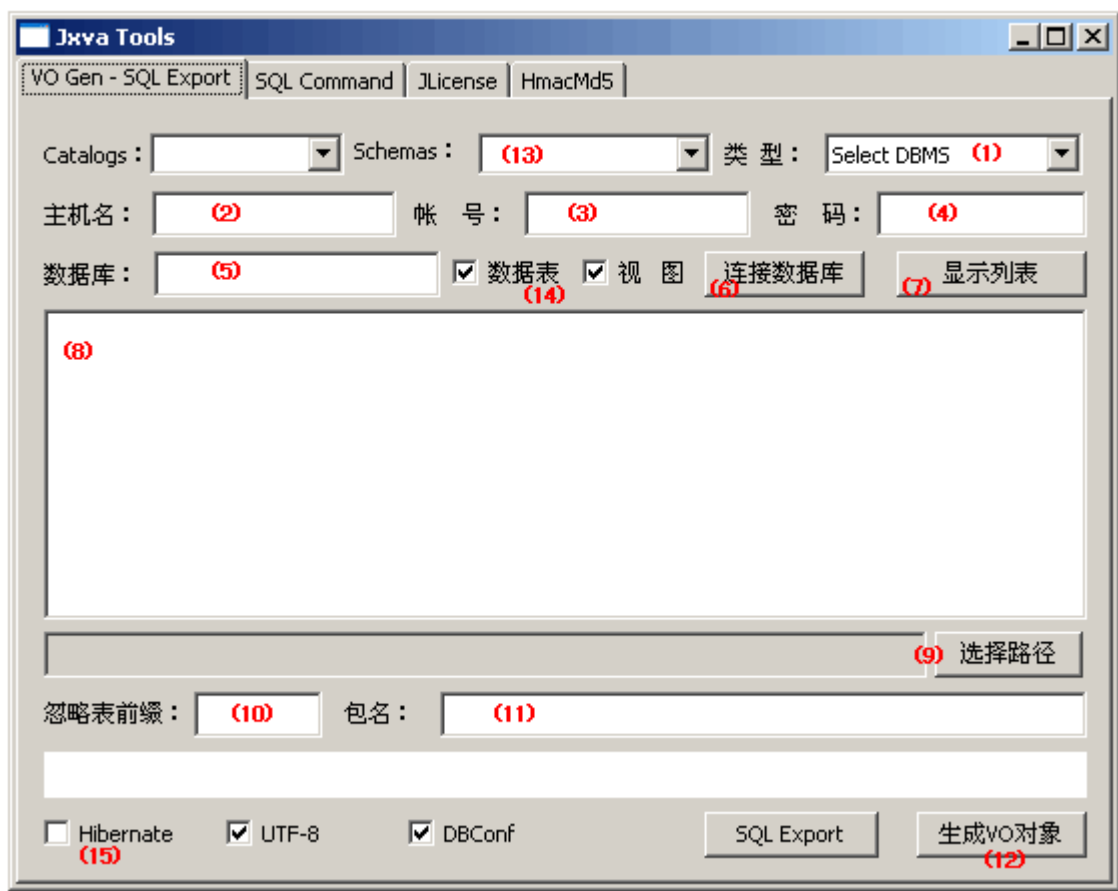



图 3

## 2.2 从源代码工程中运行

同时也可以从源代码工程中运行 Jxva Tool 自动生成工具，展开 JAVA 包到

com.jxva.tool 下， 图 4，右击 Tool.java 即可以运行，运行界面如图 3 所示。

## 2.3 利用 JxvaPlugin 插件生成

待开发...

## 2.4 自动生成工具的使用

### 2.4.1 建立数据表

这里我们以 Mysql 数据库（其它数据库类同）为例来讲解自动生成工具的使用，首先建立一个名为 demo 的数据库，同时建立三个数据表，表结构如下：

```
drop database if exists demo;
create database demo default charset=utf8;
use demo;
drop table if exists tbl_increment;
drop table if exists tbl_author;
drop table if exists tbl_book;
-- jxva framework 自增字段存储表
create table tbl_increment(
    pktb    varchar(64) not null,
    pk      integer     not null default 0,
    primary key(pktb)
)engine=innodb default charset=utf8;
-- 以下两个表是为大家学习所用的
-- 建立一个作者表
create table tbl_author(
    authorid    integer     not null default 0,
    authorname  varchar(32),
    primary key(authorid)
)engine=innodb default charset=utf8;
-- 建立一个存放书信息表
create table tbl_book(
    bookid      integer not null auto_increment,
    bookname    varchar(64),
    amount      integer default 0,
    authorid    integer not null default 0,
    primary key(bookid),
    -- 外键设为作者Id authorid,级联删除
    FOREIGN KEY (authorid) REFERENCES tbl_author (authorid) ON DELETE
    CASCADE
)engine=innodb default charset=utf8;
-- 对书名建立唯一约束
create unique index tbl_book_unique on tbl_book(bookname);
-- 建立一个作者与书数据的书信息视图
drop view if exists tbl_bookinfo;
create view tbl_bookinfo as select
tbl_author.authorid,tbl_author.authorname,tbl_book.bookid,tbl_book.bo
```

```
okname,tbl_book.amount from tbl_author,tbl_book where  
tbl_author.authorid=tbl_book.authorid;
```

其中数据表 `tbl_increment` 是存储用户自定义自动递增字段之用，下一节中我们将详细说明此自增字段表 `tbl_increment` 的用途，其它两个数据表 `tbl_author` 与 `tbl_book`，是为大家学习所用。

## 2.4.2 生成数据模型

建立数据库及数据表后，我们就可以使用 Jxva Tool 自动生成工具了，按照图 3 红色标记的操作步骤说明如下：

- (1) 选择数据库类型：对于 DB2 我们一般选择 IBM DB(jcc)模式；
- (2) 填写主机名：本机可以填写 127.0.0.1（不建议使用 localhost），远程主机可以填写 IP 或域名；
- (3) 填写数据库帐号；
- (4) 填写数据库密码；
- (5) 填写数据库名称；
- (6) 连接数据库：如果以上步骤你都填写正确并合法，你将在（信息提示区）看到连接成功提示；
- (7) 点击显示列表：在 (8) 中你将看到数据库中所有数据表与视图；
- (8) 选择数据表或视图：你可以使用鼠标单选也可以使用 Ctrl+鼠标多选；
- (9) 选择数据表模型存放路径：例如你要把生成的数据表模型放在包名为 `com.jxva.demo.model` 的包中，文件夹的实际路径为 `d:/com/jxva/demo/model`，则您应该点击(选择路径)按钮，选择 `d:/`，然后在（包名）一栏输入 `com.jxva.demo.model`，点击生成 VO 对象按钮，数据表模型、`DBConf.properties`、`hibernate.cfg.xml` 与 `hibernate.hbm.xml`（为 Hibernate 使用者提供的，Jxva DAO 并不使用该配置文件）将会生成到目录 `d:/com/jxva/demo/model` 下。
- (10) 填写忽略表前缀：如果你在建表时统一加了如 `tbl_` 的表前缀，你可以忽略掉这些表前缀，生成的数据表模型类名将更简洁；当然你也可以不填写忽略表前缀，将自动生成数据表全名的模型类名；
- (11) 填写包名：你的数据表模型如果需要包名可以直接填写你需要的包名

(强烈建议您填写) 如: com.jxva.demo.model 等;

(12) 点击生成 VO 对象;

(13) 如果你使用的数据库具备多个 Schems, 你也可对你所需要的 Schems 进行选择;

(14) 如果你只想生成表或视图, 你也可以在 (14) 中进行选择;

(15) 此步骤可以帮你同时生成 Hibernate3 HBM 文档。

如果你的步骤正确并操作成功, 将会在你选择的路径中自动生成: 数据表及视图模型 Author.java、Book.java、Bookinfo.java、Increment.java、DBConf.properites、hibernate.cfg.xml 与 hibernate.hbm.xml (如果你 (15) 进行了, 为 Hibernate 使用者提供的, Jxva DAO 并不使用该配置文件), 笔者操作的成功界面如下图:

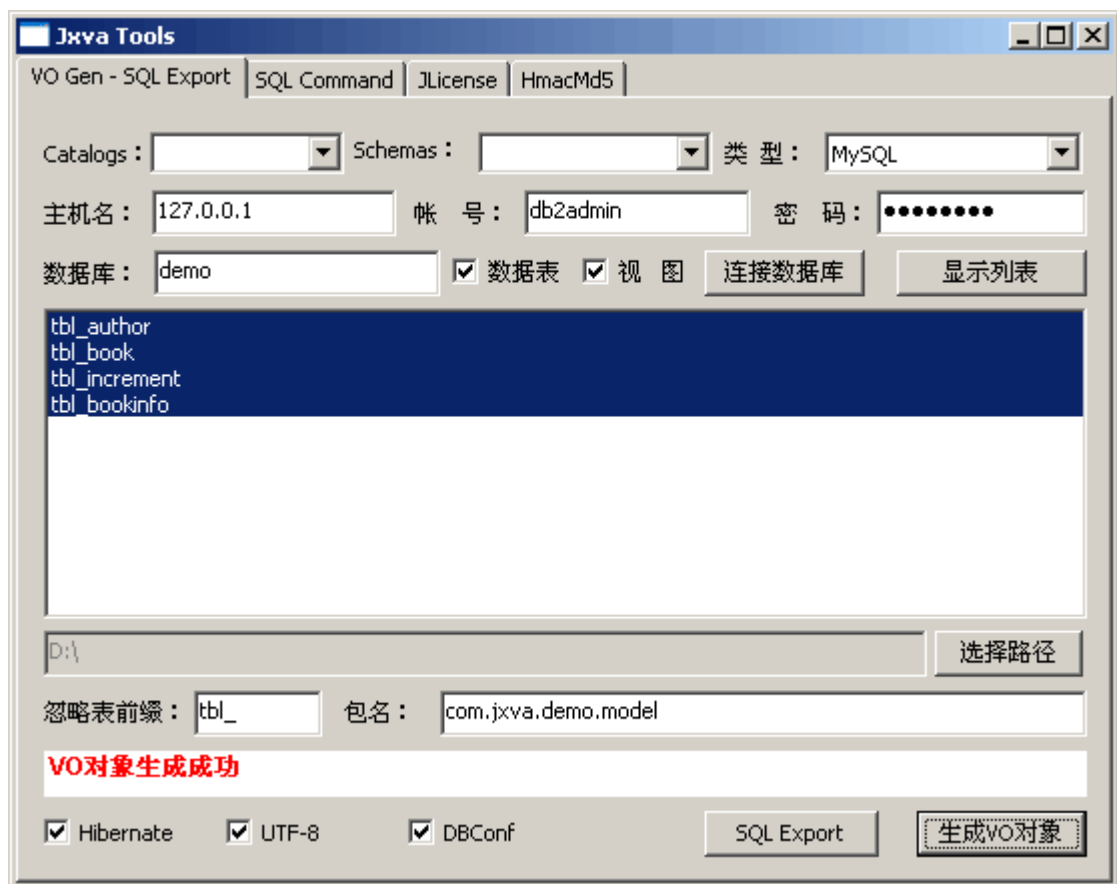


图 4

生成的数据表模型如下:

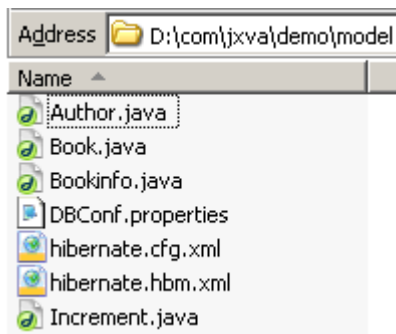


图 5

## 2.5 生成文件说明

### 2.5.1 DBConf.properties 说明

生成的 DBConf.properties 文件内容如下：（笔者加入了红色文字注释说明）

```
#DB2|SQLSERVER|MYSQL|ORACLE 支持的数据库，目前主要支持此四种
jxva.connection.dbtype = MYSQL #当前的数据库类型

#debug output sql true | false #是否输出sql语句
jxva.connection.debug = false #默认不输出，在开发模式时可以开启

#defined increment table for saving user-defined increment fields
jxva.connection.systbl= tbl_increment #自增字段表，可以自行定义表名

#proxool|c3p0|dbcp|jndi #采用的数据库连接池，目前支持此四种，读者也可以进行扩展
jxva.connection.pool = proxool #默认为proxool

jxva.connection.username = db2admin #数据库用户名
jxva.connection.password = pass1009 #数据库密码

jxva.connection.max_size = 50 #连接池最大连接数
jxva.connection.min_size = 10 #连接池最小连接数

jxva.connection.asia = Jxva #连接映射，proxool需要用到
jxva.connection.jndi = java:comp/env/jdbc/test #jndi接口

jxva.connection.driver_class = com.mysql.jdbc.Driver #数据库驱动
jxva.connection.url =
jdbc:mysql://127.0.0.1:3306/demo?zeroDateTimeBehavior=convertToNull&useUnicode=true&characterEncoding=utf-8 #数据库连接串

#DB2 DB2连接样例
```

```

#jxva.connection.driver_class = com.ibm.db2.jcc.DB2Driver
#jxva.connection.url = jdbc:db2://127.0.0.1:50000/test

#MYSQL MySQL连接样例
#jxva.connection.driver_class = com.mysql.jdbc.Driver
#jxva.connection.url =
jdbc:mysql://127.0.0.1:3306/test?zeroDateTimeBehavior=convertToNull&useUnicode=true&characterEncoding=utf-8

#MSSQL MSSQL连接样例，采用的是MSSQL2005的驱动，同时适用于MSSQL2000
#jxva.connection.driver_class =
com.microsoft.sqlserver.jdbc.SQLServerDriver
#jxva.connection.url =
jdbc:sqlserver://127.0.0.1:1433;DatabaseName=test;SelectMethod=direct

#ORACLE ORACLE连接样例
#jxva.connection.driver_class = oracle.jdbc.driver.OracleDriver
#jxva.connection.url = jdbc:oracle:thin:@127.0.0.1:1521:test

```

DBConf.properites 由自动生成工具生成，开发人员基本无需修改即可使用，一般此文件存放于 WEB-INF/classes 或类路径根下面，当然也可以存放于其它类路径下面，具体笔者将在 DAO 框架的 DAOFactory 里面详细说明。

## 2.5.2 数据模型文件说明

其中生成的 Author.java、Book.java、Bookinfo.java、Increment.java 我们称为数据表及视图模型（有些人也称为 POJO，VO，BO 等，笔者统称为数据模型）。

### (1) Increment.java 内容如下：

```

package com.jxva.demo.model;

import java.io.Serializable;
import com.jxva.dao.Table;

/**
 *
 * auto generate by jxva framework
 * on 2008-06-16 01:25:39
 *
 */

```



```

@Table(name="tbl_increment",increment="null",primarykeys={"pktb"})
public class Increment implements Serializable{

    private java.lang.String pktb;
    private java.lang.Integer pk;

    public java.lang.String getPktb(){
        return this.pktb;
    }
    public void setPktb(java.lang.String pktb){
        this.pktb=pktb;
    }

    public java.lang.Integer getPk(){
        return this.pk;
    }
    public void setPk(java.lang.Integer pk){
        this.pk=pk;
    }

    public boolean equals(Object obj){
        return super.equals(obj);
    }

    public int hashCode(){
        return super.hashCode();
    }

    public String toString(){
        return super.toString();
    }

}

```

为了解决hibernate中的XML文档的复杂与难以理解的配置，笔者在数据模型中新引入的Java5注解元素@Table，在Jxva DAO 主要表示此java文件对应的是一个数据表，其中表名name为tbl\_increment，无自动递增字段，主键字段是pktb，其它数据都是读者熟悉的getter与setter方法，在此就不再说明了。

## (2) Author.java文件内容如下：

```

package com.jxva.demo.model;
import java.io.Serializable;
import com.jxva.dao.Table;
@Table(name="tbl_author",increment="null",primarykeys={"authorid"})

```

```
public class Author implements Serializable{
    private java.lang.Integer authorid;
    private java.lang.String authorname;
    //...为了减少内容篇幅，笔者在此将getter与setter方法都已省略
}
```

(3) Book.java文件内容如下：

```
package com.jxva.demo.model;
import java.io.Serializable;
import com.jxva.dao.Table;
@Table(name="tbl_book",increment="bookid",primarykeys={"bookid"})
public class Book implements Serializable{
    private java.lang.Integer bookid;
    private java.lang.String bookname;
    private java.lang.Integer amount;
    private java.lang.Integer authorid;
    //...为了减少内容篇幅，笔者在此将getter与setter方法都已省略
}
```

(4) Bookinfo.java文件内容如下：

```
package com.jxva.demo.model;
import java.io.Serializable;
import com.jxva.dao.Table;
@Table(name="tbl_bookinfo",increment="bookid",primarykeys={"authorid"})
public class Bookinfo implements Serializable{
    private java.lang.Integer authorid;
    private java.lang.String authorname;
    private java.lang.Integer bookid;
    private java.lang.String bookname;
    private java.lang.Integer amount;
    //...为了减少内容篇幅，笔者在此将getter与setter方法都已省略
}
```

Bookinfo.java对应是一个视图，其中的increment="bookid"与primarykeys={"authorid"}由系统自动生成，可以不用修改，在实际编程中没有用到。

## 2.5.3 Hibernate 相关 xml 文档

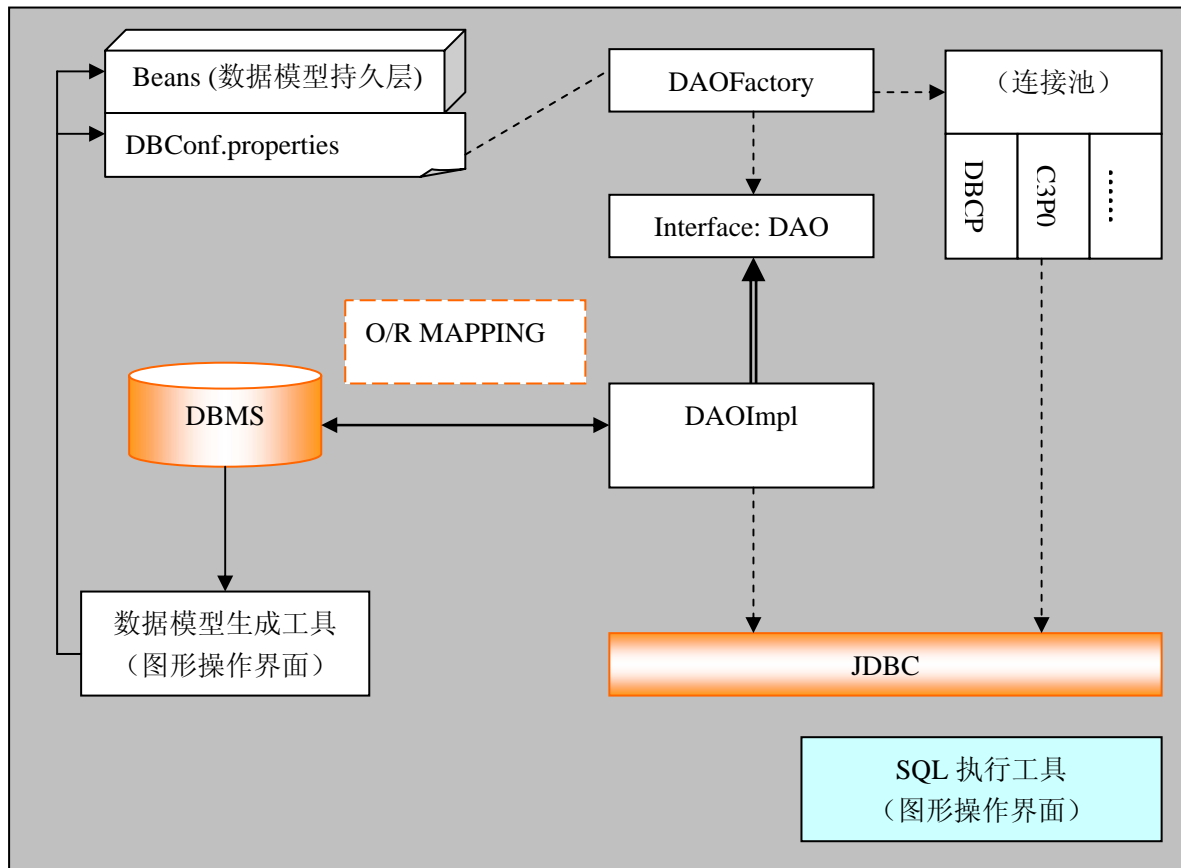
生成的hibernate.cfg.xml与hibernate.hbm.xml主要方便hibernate的开发人员使用，在Jxva DAO中没有用到，在此就不再详说。

数据模型生成之后，在下一章我们将带您步入简单、快捷、高性能的Jxva DAO开发中。

### 3、DAO 框架

#### 3.1 DAO 整体架构及数据支持

Jxva DAO 的整体架构如下图所示：



DAO 目前支持的数据库系统：MySQL、SQLServer、Oracle、DB2（也可以支持别的数据库系统，需要加入相应数据库方言）

DAO 支持的字段类型：字符串类型 INTEGER、BIGINT、SMALLINT、LONGVARCHAR、CHAR、VARCHAR、CLOB、TIMESTAMP、DATE、REAL、FLOAT、NUMERIC、DECIMAL、BLOB、LONGVARBINARY、DOUBLE（也可能支持别的字段类型，但没有经过完全测试）

#### 3.2 创建 DAOFactory

DAOFactory factory=null;// DAOFactory 是线程安全的

//第一个参数即生成的配置文件 DBConf.properties 所位于的包(文件的后缀名.properties 不要写)

- factory=DAOFactory.getInstance("com.jxva.demo.model.DBConf");

//如果 DBConf.properties 存放于类路径根目录，可以使用下面的方法，笔者强烈建议采用如下方式（因为 Jxva MVC 框架中自动数据连接关闭将使用此方式）

- factory=DAOFactory.getInstance();

### 3.3 创建 DAO

利用上一节中的 DAOFactory，我们可以非常简单的创建 DAO：

```
DAO dao=factory.createDAO();
```

### 3.4 关闭 DAO

当你不再使用 dao 或是捕捉到异常时，调用 dao.close() 方法关闭 dao，本人认为一个合格的程序员应该在适当的时候关闭 DAO。

如果读者结合使用 Jxva MVC，DAOFactory 将不用创建，同时 DAO 也不用创建与关闭，完全由 Jxva MVC 框架自动完成，在 MVC 框架中我们将会更深入的讲解。

DAOFactory 的关闭主要是在进行多个不同类型的数据库操作时，当整个 DAO 操作过程完成之后进行关闭，可以调用 factory.close()方法，仅在一个数据库系统中，可以不用关闭。

一个完整使用 DAOFactory 与 DAO 的程序如下：

```
import com.jxva.dao.DAO;
import com.jxva.dao.DAOFactory;
public class DaoStudy {
    private static DAOFactory factory;
    public static void main(String[] args) {
        factory=DAOFactory.getInstance();
        DAO dao=factory.createDAO();
        //dao操作
        dao.close();
    }
}
```

## 3.5 DAO 的操作

### 3.5.1 插入操作

(1) 使用系统自增数据表机制或自定义字段值

原型: `public void insert(Object obj);`

实例:

```
Author author=new Author();
```

```
//A使用默认主键自增
```

```
author.setAuthorid(dao.getAutoid(Author.class).intValue());
```

```
//B指定自增字段, 在此指定的递增字段为authorid
```

```
//author.setAuthorid(dao.getAutoid(Author.class,"authorid").intValue());
```

```
//C自定义自增字段值(不推荐)
```

```
//author.setAuthorid(10);
```

```
author.setAuthorname("Martin Fowler");
```

```
dao.insert(author);
```

(2) 使用数据库自身的递增字段

原型: `public void insert(Object obj);`

实例:

```
Book book=new Book();
```

```
//这里我们设置了bookid的值, 由于我们在建立数据表时设置了bookid为自增
```

```
字段, dao将不会对bookid做任何操作
```

```
book.setBookid(3);
```

```
book.setAmount(3);
```

```
book.setAuthorid(10);
```

```
book.setBookname("Refactoring");
```

```
dao.insert(book);
```

采用(1)中的A, B方式DAO框架将会调用自增数据表, 我们前面定义的自增数据表为tbl\_increment, 因此在读者采用此方式时, 必须建立一个自增数据表, 同时自增数据表结构必须如下定义:

<code>pktb    varchar(64) not null,</code>
--

```
pk      integer      not null default 0,
primary key(pktb)
```

### 3.5.2 更新操作

#### (1) 根据主键更新对象

原型: `public void updateByPrimaryKey(Object obj);`

实例:

//将主键authorid为1的authorname更改为"Martin Fowler"

```
Author author=new Author();
author.setAuthorid(1);
author.setAuthorname("Martin Fowler");
dao.updateByPrimaryKey (author);
```

#### (2) 自定义条件批量更新

原型: `public void update(Object obj, String where);`

实例:

//将authorid小于10的authorname批量更改为"Martin Fowler"

```
Author author=new Author();
author.setAuthorname("Martin Fowler");
dao.update (author, "authorid<10");
```

### 3.5.3 删除操作

#### (1) 根据主键删除对象

原型: `public <T> void deleteByPrimaryKey(T obj);`

实例:

//将主键authorid为1对象删除

```
Author author=new Author();
author.setAuthorid(1);
dao.deleteByPrimaryKey(author);
```

#### (2) 自定义条件批量删除

原型: `public <T> void delete(Class<T> cls,String where);`

实例:

```
//将authorid小于5并且authorname为"Martin Fowler"的对象全部删除
dao.delete (Author.class, "authored<5 and authorname='Martin Fowler'");
```

### 3.5.4 单个查询

#### (1) 根据主键进行单个查询

原型: `public <T> T findByPrimaryKey(T obj);`

实例:

//查询主键authorid为1对象

```
Author author=new Author();
```

```
author.setAuthorid(1);
```

//由于DAO框架使用了泛型，所以查询的返回结果直接为Author，不用强制转换

```
Author result=dao.findByPrimaryKey(author);
```

#### (2) 自定义查询条件单个查询

原型: `public <T> T findByPrimaryKey(T obj);`

实例:

//查询authorname为"Martin Fowler"的对象

```
Author result=dao.findSingle(Author.class, "authorname='Martin Fowler'");
```

### 3.5.5 单表查询

#### (1) 全表查询

原型: `public <T> T findByPrimaryKey(T obj);`

实例:

//查询所有author对象

```
List<Author> authors=dao.find(Author.class);
```

//这里的 List<Author>也是由于 DAO 框架采用泛型而不用强制转换（在以后的章节中，你将会见到更多采用泛型的方式）

#### (2) 自定义条件单表查询

原型: `public <T> T findSingle(Class<T> cls, String where);`

实例:



//查询所有 authorname 为"Martin Fowler"的对象

```
List<Author> authors=dao.find(Author.class, "authorname='Martin Fowler'");
```

我们也可以利用此方法实现全表查询，如下：

A、查询条件为 null

```
List<Author> authors=dao.find(Author.class,null);
```

B、查询条件为''

```
List<Author> authors=dao.find(Author.class, '');
```

### (3) 自定义条件单表分页查询

原型：**public** <T> List<T> find(Class<T> cls,String where,**int** pagesize,**int** currentpage);

实例：

//查询 authorid 小于 100 的对象并分页

```
List<Author> authors=dao.find(Author.class, "authoreid<100",20,3);
```

其中

参数 20：表示每页显示条数

参数 3：表示当前为第 3 页，返回的结果为第 3 页的对象

### (4) 视图的查询

视图的查询完全与数据表的查询一样，例如：

//查询 authorid 小于 100 的所有 book 信息

```
List<Bookinfo> bookinfos=dao.find(BookInfo.class, "authoreid<100");
```

其他的视图查询在此就不再重复说明。

所有查询条件都不包含"where"，同时查询条件不区分大小写，我们强烈建议读者使用小写，查询条件中的字段完全对应于数据模型中的字段。

返回结果 results 中的每一个元素为与数据库表对应的数据模型的对象，上例中，即为 Author，可以用两种方式遍历查询结果：

```
// 方式一
for(int i=0;i<authors.size();i++){
    Author author=authors.get(i);
}

//方式二
```

```
for (Author author: authors) {  
    //...  
}
```

### 3.5.6 多表查询

#### (1) 自定义条件多表查询

原型: `public <T> List<T> find(Class<T>[] cls, String where);`

实例:

// 查询authorname为"Martin Fowler"的所有book 信息

#### (2) 自定义条件多表分页查询

原型: `public <T> List<T> find(Class<T>[] cls, String where, int  
pagesize, int currentpage);`

实例:

// 分页查询authorname为"Martin Fowler"的所有book 信息

### 3.5.7 事务操作

`dao.beginTransaction();` //开启事务

`//其它 dao 操作.....`

`dao.commit();` //如果成功, 则提交

`//dao.rollback();` //如果失败, 则回滚

### 3.5.8 使用 SQL

原型: `public void execute(String sql);`

实例:

### **3.5.9 其它操作**

## **3.6 与 Hibernate 的比较**

## 4、框架核心配置

<http://code.google.com/p/jxva/downloads/>

## 5、MVC 框架

<http://code.google.com/p/jxva/downloads/>

## 6、Tag 标签

<http://code.google.com/p/jxva/downloads/>

## 7、RBAC 体系

<http://code.google.com/p/jxva/downloads/>

## 8、OXM 框架