

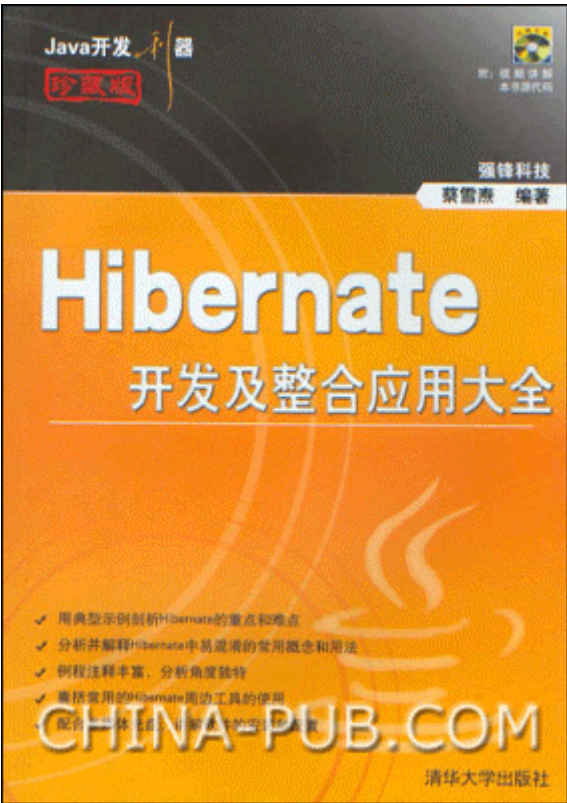
《Hibernate开发及整合应用大全》 【作者】 蔡雪燕

清华大学出版社

<http://www.china-pub.com/computers/common/info.asp?id=25602>

<http://www.huachu.com.cn/itbook/itbookinfo.asp?lbbh=BB071045563>

作者 blog：<http://www.blogjava.net/caixuetao>



目录

上篇 Hibernate 技术

第一章 对象持久化.....	错误！未定义书签。
1.1 JDBC面临的一些问题.....	错误！未定义书签。
1.1.1 JDBC简介.....	错误！未定义书签。
1.1.2 繁琐的代码问题.....	错误！未定义书签。
1.1.3 多表连接问题.....	错误！未定义书签。
1.1.4 表间级联问题.....	错误！未定义书签。
1.1.5 层与层之间的耦合严重.....	错误！未定义书签。
1.1.6 性能问题.....	错误！未定义书签。
1.2 使用ORM.....	错误！未定义书签。
1.3 软件分层体系结构.....	错误！未定义书签。

1.4	域模型.....	错误！未定义书签。
1.4.1	实体域对象.....	错误！未定义书签。
1.4.2	过程域对象.....	错误！未定义书签。
1.5	小结.....	错误！未定义书签。
第二章	开始Hibernate之旅.....	错误！未定义书签。
2.1	Hibernate简介.....	错误！未定义书签。
2.2	搭建开发环境.....	错误！未定义书签。
2.2.1	安装Access数据库.....	错误！未定义书签。
2.2.2	下载Hibernate.....	错误！未定义书签。
2.2.3	安装和配置JDK.....	错误！未定义书签。
2.2.3	下载安装UltraEdit编辑器并进行设置.....	错误！未定义书签。
2.3	第一个使用Hibernate操作数据库的例子.....	错误！未定义书签。
2.3.1	在Access中创建新库及表.....	错误！未定义书签。
2.3.2	编写Java文件(持久化类)Student.java.....	错误！未定义书签。
2.2.3	把hibernate的包解压到当前路径.....	错误！未定义书签。
2.2.4	编写取得session的类HibernateUtil.java.....	错误！未定义书签。
2.2.5	编写操作数据库的Java文件DAOImp.java.....	错误！未定义书签。
2.2.6	对Student.java文件创建一个Hibernate映射文件Student.hbm.xml.....	错误！未定义书签。
2.2.7	配置Hibernate的描述文件Hibernate.cfg.xml.....	错误！未定义书签。
2.2.8	编写业务逻辑处理文件BM.java.....	错误！未定义书签。
2.2.9	运行测试文件BM.java并查看Access数据库中的结果.....	错误！未定义书签。
2.2.10	创建学生对象的运行时序图.....	错误！未定义书签。
2.2.11	常见问题解决.....	错误！未定义书签。
2.3	小结.....	错误！未定义书签。
第三章	Hibernate基础.....	错误！未定义书签。
3.1	Hibernate配置相关的类.....	错误！未定义书签。
3.1.1	Configuration类(负责管理Hibernate的配置信息).....	错误！未定义书签。
3.1.2	SessionFactory类(负责创建Session实例).....	错误！未定义书签。
3.2	Session类.....	错误！未定义书签。
3.2.1	使用threadLocal变量.....	错误！未定义书签。
3.3.2	Session的缓存.....	错误！未定义书签。
3.2	对象在JVM中的生命周期.....	错误！未定义书签。
3.3	对象在Hibernate中的状态.....	错误！未定义书签。
3.3.1	瞬时(Transient)状态.....	错误！未定义书签。
3.3.2	持久化状态.....	错误！未定义书签。
3.3.3	脱管状态.....	错误！未定义书签。
3.4	对象识别.....	错误！未定义书签。
3.4.1	以内存地址识别.....	错误！未定义书签。
3.4.2	以对象携带的信息识别.....	错误！未定义书签。
3.5	对象关联关系.....	错误！未定义书签。
3.5.1	一对一关联.....	错误！未定义书签。
3.5.2	一对多关联.....	错误！未定义书签。
3.5.3	多对多关联.....	错误！未定义书签。

3.5.4	对象级联 ( cascade ) 关系 .....	错误！未定义书签。
3.6	SQL语言数据查询基础 .....	错误！未定义书签。
3.6.1	简单查询 .....	错误！未定义书签。
3.6.2	连接查询 .....	错误！未定义书签。
3.6.3	子查询 .....	错误！未定义书签。
3.6.4	联合查询 .....	错误！未定义书签。
3.6.5	报表查询 .....	错误！未定义书签。
3.7	Hibernate数据检索策略 .....	错误！未定义书签。
3.7.1	立即检索 .....	错误！未定义书签。
3.7.2	延迟检索 .....	错误！未定义书签。
3.8	预先抓取 .....	错误！未定义书签。
3.8.1	对象之间全都是立即加载 .....	错误！未定义书签。
3.8.2	对象之间全都是预先抓取 .....	错误！未定义书签。
3.8.3	外连接与预先抓取的区别 .....	错误！未定义书签。
3.8.4	总结 .....	错误！未定义书签。
3.9	批量加载 .....	错误！未定义书签。
3.9.1	批量立即加载 .....	错误！未定义书签。
3.9.2	批量延迟加载 .....	错误！未定义书签。
3.10	对预先抓取的限制 .....	错误！未定义书签。
3.10.1	Hibernate2.x的限制 .....	错误！未定义书签。
3.10.2	在一条SQL语句中预先抓取多个集合 .....	错误！未定义书签。
3.9	小结 .....	错误！未定义书签。
第四章	操纵实体对象 .....	错误！未定义书签。
4.1	编写持久化类(Persistent Classes) .....	错误！未定义书签。
4.1.1	分析已知的XXX.hbm.xml文件 .....	错误！未定义书签。
4.1.2	编写持久化类 .....	错误！未定义书签。
4.2	session的保存、删除、更新方法 .....	错误！未定义书签。
4.2.1	save()方法 .....	错误！未定义书签。
4.2.2	update()方法 .....	错误！未定义书签。
4.2.3	saveOrUpdate()方法 .....	错误！未定义书签。
4.2.4	delete()方法 .....	错误！未定义书签。
4.3	通过主键id取得数据对象 .....	错误！未定义书签。
4.3.1	get()方法 .....	错误！未定义书签。
4.3.2	load()方法 .....	错误！未定义书签。
4.4	Query接口 .....	错误！未定义书签。
4.4.1	绑定参数 .....	错误！未定义书签。
4.4.2	使用命名查询 ( namedQuery ) .....	错误！未定义书签。
4.4.3	list()方法 .....	错误！未定义书签。
4.4.4	uniqueResult()方法 .....	错误！未定义书签。
4.4.5	iterator()方法 .....	错误！未定义书签。
4.4.6	查询缓存 .....	错误！未定义书签。
4.5	清除缓存对象 .....	错误！未定义书签。
4.5.1	clear()方法 .....	错误！未定义书签。
4.5.2	evict()方法 .....	错误！未定义书签。

4.6	cascade属性的用法 .....	错误！未定义书签。
4.6.1	none .....	错误！未定义书签。
4.6.2	save-update .....	错误！未定义书签。
4.6.3	delete .....	错误！未定义书签。
4.6.4	delete-orphan .....	错误！未定义书签。
4.7	级联持久化临时对象 .....	错误！未定义书签。
4.7.1	根对象为临时对象 .....	错误！未定义书签。
4.7.2	根对象为持久对象 .....	错误！未定义书签。
4.7.3	根对象为脱管对象 .....	错误！未定义书签。
4.8	小结 .....	错误！未定义书签。
第五章	使用关联关系操纵对象 .....	错误！未定义书签。
5.1	一对一关联关系的使用 .....	错误！未定义书签。
5.1.1	以主键关联 .....	错误！未定义书签。
5.1.2	以外键关联 .....	错误！未定义书签。
5.1.3	默认的级联关系 .....	错误！未定义书签。
5.1.4	延迟加载 .....	错误！未定义书签。
5.2	一对多、多对一关联关系的使用 .....	错误！未定义书签。
5.2.1	单向关联 .....	错误！未定义书签。
5.2.2	双向关联 .....	错误！未定义书签。
5.3	多对多关联关系的使用 .....	错误！未定义书签。
5.3.1	添加关联关系 .....	错误！未定义书签。
5.3.2	删除关联关系 .....	错误！未定义书签。
5.4	小结 .....	错误！未定义书签。
第六章	Hibernate数据查询 .....	错误！未定义书签。
6.1	HQL检索方式 .....	错误！未定义书签。
6.1.1	最简单的查询 .....	错误！未定义书签。
6.1.2	属性查询 .....	错误！未定义书签。
6.1.3	实例化查询结果 .....	错误！未定义书签。
6.1.4	连接查询 .....	错误！未定义书签。
6.1.5	统计函数查询 .....	错误！未定义书签。
6.1.6	集合过滤 .....	错误！未定义书签。
6.1.7	子查询 .....	错误！未定义书签。
6.1.8	多态查询 .....	错误！未定义书签。
6.2	QBC检索方式 .....	错误！未定义书签。
6.2.1	常用的限定方法 .....	错误！未定义书签。
6.2.2	连接限定 .....	错误！未定义书签。
6.2.3	动态查询 .....	错误！未定义书签。
6.2.4	QBE查询方式 .....	错误！未定义书签。
6.2.5	分页查询 .....	错误！未定义书签。
6.2.6	DetachedCriteria .....	错误！未定义书签。
6.3	使用本地SQL检索 .....	错误！未定义书签。
6.3.1	创建一个基于SQL的Query .....	错误！未定义书签。
6.3.2	别名和属性引用 .....	错误！未定义书签。
6.3.3	命名SQL查询 .....	错误！未定义书签。

6.3.4	自定义insert、update、delete语句	错误！未定义书签。
6.4	小结	错误！未定义书签。
第七章	XML基础	错误！未定义书签。
7.1	XML基本概念	错误！未定义书签。
7.1.1	XML的用途	错误！未定义书签。
7.1.2	结构化	错误！未定义书签。
7.1.3	XML元素	错误！未定义书签。
7.1.4	XML属性	错误！未定义书签。
7.1.5	XML数据岛	错误！未定义书签。
7.1.6	XML命名空间	错误！未定义书签。
7.2	XML文档	错误！未定义书签。
7.2.1	Well-formed XML(良好格式的XML)	错误！未定义书签。
7.2.2	Valid XML(有效的XML)	错误！未定义书签。
7.3	DTD文档	错误！未定义书签。
7.3.1	内部DTD	错误！未定义书签。
7.3.2	外部DTD	错误！未定义书签。
7.4	解析器	错误！未定义书签。
7.4.1	DOM	错误！未定义书签。
7.4.2	SAX	错误！未定义书签。
7.5	小结	错误！未定义书签。
第八章	Hibernate配置	错误！未定义书签。
8.1	配置数据库连接	错误！未定义书签。
8.1.1	在非管环境中	错误！未定义书签。
8.1.2	在受管理环境中	错误！未定义书签。
8.2	配置事务	错误！未定义书签。
8.3	小结	错误！未定义书签。
第九章	对象—关系映射配置	错误！未定义书签。
9.1	类映射	错误！未定义书签。
9.2	Hibernate的主键策略	错误！未定义书签。
9.2.1	Hibernate对主键id赋值	错误！未定义书签。
9.2.2	应用程序自己对id赋值	错误！未定义书签。
9.2.3	由数据库对id赋值	错误！未定义书签。
9.3	使用复合id(composite-id)	错误！未定义书签。
9.3.1	不把复合主键封装成类	错误！未定义书签。
9.3.2	把复合主键封装成类	错误！未定义书签。
9.3.3	复合主键中的字段与其他类具有关联关系	错误！未定义书签。
9.4	属性映射	错误！未定义书签。
9.4.1	基本值类型	错误！未定义书签。
9.4.2	映射Blob、Clob	错误！未定义书签。
9.5	组件（Component）映射	错误！未定义书签。
9.5.1	普通的组件映射	错误！未定义书签。
9.5.2	集合组件的映射	错误！未定义书签。
9.6	自定义数据类型	错误！未定义书签。
9.6.1	使用UserType	错误！未定义书签。

9.6.2 使用CompositeUserType.....	错误！未定义书签。
9.7 继承关系的映射.....	错误！未定义书签。
9.7.1 每一个具体类对应一张数据表.....	错误！未定义书签。
9.7.2 一张表对应一整棵类继承树.....	错误！未定义书签。
9.7.3 一个类对应一张表.....	错误！未定义书签。
9.7.4 比较映射策略.....	错误！未定义书签。
9.8 小结.....	错误！未定义书签。
第十章 集合映射.....	错误！未定义书签。
10.1 映射Map.....	错误！未定义书签。
10.1.1 HashMap.....	错误！未定义书签。
10.1.2 LinkedHashMap.....	错误！未定义书签。
10.1.3 TreeMap.....	错误！未定义书签。
10.1.4 在Hibernate中映射Map.....	错误！未定义书签。
10.2 映射Set.....	错误！未定义书签。
10.2.1 HashSet.....	错误！未定义书签。
10.2.2 LinkedHashSet.....	错误！未定义书签。
10.2.3 TreeSet.....	错误！未定义书签。
10.2.4 在Hibernate中映射Set.....	错误！未定义书签。
10.3 映射List.....	错误！未定义书签。
10.3.1 ArrayList.....	错误！未定义书签。
10.3.2 LinkedList.....	错误！未定义书签。
10.3.3 在Hibernate中映射List.....	错误！未定义书签。
10.4 映射Bag.....	错误！未定义书签。
10.4.1 <bag>标签.....	错误！未定义书签。
10.4.2 <idbag>标签.....	错误！未定义书签。
10.5 集合的排序.....	错误！未定义书签。
10.5.1 在数据库中排序.....	错误！未定义书签。
10.5.2 在内存中排序.....	错误！未定义书签。
10.6 小结.....	错误！未定义书签。
第十一章 Hibernate事务与Cache管理.....	错误！未定义书签。
11.1 Hibernate事务管理.....	错误！未定义书签。
11.1.1 事务介绍.....	错误！未定义书签。
11.1.2 什么是事务隔离级别.....	错误！未定义书签。
11.1.3 选择合适的隔离级别.....	错误！未定义书签。
11.1.4 设置隔离级别.....	错误！未定义书签。
11.1.5 事务中对操作的flush()函数调用.....	错误！未定义书签。
11.1.6 使用JDBC事务.....	错误！未定义书签。
11.1.7 使用JTA事务.....	错误！未定义书签。
11.1.8 小结.....	错误！未定义书签。
11.2 Hibernate中对数据的锁定.....	错误！未定义书签。
11.2.1 悲观锁（Pessimistic Locking）的用法.....	错误！未定义书签。
11.2.2 乐观锁（Optimistic Locking）的用法.....	错误！未定义书签。
11.2.3 Session的lock()与update()方法.....	错误！未定义书签。
11.3 Cache管理.....	错误！未定义书签。

11.3.1	Hibernate中的Cache .....	错误！未定义书签。
11.3.2	在Hibenate中运用Ehcache .....	错误！未定义书签。
11.4	小结 .....	错误！未定义书签。

## 下篇 Hibernate 与其他工具整合应用

第十二章	JSP技术的应用 .....	错误！未定义书签。
12.1	JSP技术概述 .....	错误！未定义书签。
12.1.1	什么是JSP .....	错误！未定义书签。
12.1.2	动态网页技术比较 .....	错误！未定义书签。
12.1.3	为何使用JSP .....	错误！未定义书签。
12.2	搭建JSP开发环境 .....	错误！未定义书签。
12.2.1	安装JDK .....	错误！未定义书签。
12.2.2	JDK的配置 .....	错误！未定义书签。
12.2.3	安装和配置Tomcat .....	错误！未定义书签。
12.2.4	测试第一个JSP页面 .....	错误！未定义书签。
12.3	JSP核心语法 .....	错误！未定义书签。
12.3.1	指令元素 .....	错误！未定义书签。
12.3.2	脚本元素 .....	错误！未定义书签。
12.3.3	动作元素 .....	错误！未定义书签。
12.3.4	注释 .....	错误！未定义书签。
12.4	JSP的内置对象 .....	错误！未定义书签。
12.4.1	out对象 .....	错误！未定义书签。
12.4.2	request对象 .....	错误！未定义书签。
12.4.3	response对象 .....	错误！未定义书签。
12.4.4	session对象 .....	错误！未定义书签。
12.4.5	application对象 .....	错误！未定义书签。
12.4.6	pageContext对象 .....	错误！未定义书签。
12.5	使用JavaBean .....	错误！未定义书签。
12.5.1	JavaBean的属性 .....	错误！未定义书签。
12.5.2	JavaBean的方法编写 .....	错误！未定义书签。
12.5.3	在JSP中使用JavaBean .....	错误！未定义书签。
12.6	使用Servlet .....	错误！未定义书签。
12.6.1	Servlet简介 .....	错误！未定义书签。
12.6.2	Servlet的接口 .....	错误！未定义书签。
12.6.3	Servlet配置 .....	错误！未定义书签。
12.6.4	使用Servlet .....	错误！未定义书签。
12.7	JSP开发中的常用技巧 .....	错误！未定义书签。
12.7.1	同一用户不同页面之间的数据共享 .....	错误！未定义书签。
12.7.2	不同用户之间的数据共享 .....	错误！未定义书签。
12.7.3	创建错误处理页面 .....	错误！未定义书签。
12.7.4	中文乱码问题 .....	错误！未定义书签。
12.8	小结 .....	错误！未定义书签。

第十三章	MySql入门及Hibernate整合 .....	错误！未定义书签。
13.1	MySQL数据库简介.....	错误！未定义书签。
13.1.1	什么是MySQL.....	错误！未定义书签。
13.1.2	为何选用MySQL.....	错误！未定义书签。
13.2	MySQL下载及安装配置.....	错误！未定义书签。
13.2.1	下载及注意事项.....	错误！未定义书签。
13.2.2	安装配置.....	错误！未定义书签。
13.3	在字符界面使用MySQL数据库.....	错误！未定义书签。
13.3.1	启动MySQL数据库.....	错误！未定义书签。
13.3.2	连接MySQL.....	错误！未定义书签。
13.3.3	使用一条简单的查询语句.....	错误！未定义书签。
13.3.4	在一个命令行中输入多条语句.....	错误！未定义书签。
13.3.5	一条语句跨越多个命令行.....	错误！未定义书签。
13.3.6	取消语句的执行.....	错误！未定义书签。
13.3.7	MySQL命令行的常见状态.....	错误！未定义书签。
13.3.8	打开数据库的命令.....	错误！未定义书签。
13.3.9	查看数据库的命令（ show databases ） .....	错误！未定义书签。
13.3.10	查看库中数据表的语句（ show tables ） .....	错误！未定义书签。
13.3.11	查看数据表详细结构（ describe databaseName ） .....	错误！未定义书签。
13.3.12	新建数据库.....	错误！未定义书签。
13.3.13	新建表.....	错误！未定义书签。
13.3.14	插入或删除表中数据.....	错误！未定义书签。
13.3.15	删除表.....	错误！未定义书签。
13.3.16	删除数据库.....	错误！未定义书签。
13.3.17	创建新用户并给予权限.....	错误！未定义书签。
13.3.18	更改MySQL用户密码.....	错误！未定义书签。
13.3.19	从SQL文件导入数据表 .....	错误！未定义书签。
13.3.20	将文本数据转到数据库中.....	错误！未定义书签。
13.3.21	备份和恢复数据库.....	错误！未定义书签。
13.3.22	退出MySQL连接.....	错误！未定义书签。
13.3.23	关闭MySQL服务.....	错误！未定义书签。
13.4	在图形界面使用MySQL.....	错误！未定义书签。
13.4.1	打开MySQL Query Browser程序 .....	错误！未定义书签。
13.4.2	创建新库.....	错误！未定义书签。
13.4.3	新建一个数据表.....	错误！未定义书签。
13.4.4	操作表中数据.....	错误！未定义书签。
13.5	MySQL与Hibernate的整合使用 .....	错误！未定义书签。
13.5.1	编写实体对象文件.....	错误！未定义书签。
13.5.2	编写实体映射文件Student.hbm.xml .....	错误！未定义书签。
13.5.3	编写Hibernate配置文件hibernate.cfg.xml .....	错误！未定义书签。
13.5.4	写一个测试文件BM.java.....	错误！未定义书签。
13.6	小结.....	错误！未定义书签。
第十四章	Tomcat应用及Hibernate整合 .....	错误！未定义书签。
14.1	Tomcat简介 .....	错误！未定义书签。



14.1.1	什么是Tomcat.....	错误！未定义书签。
14.1.2	Tomcat与Servlet容器.....	错误！未定义书签。
14.2	安装和配置Tomcat所需资源.....	错误！未定义书签。
14.2.1	下载及安装Tomcat.....	错误！未定义书签。
14.2.2	Tomcat目录结构.....	错误！未定义书签。
14.2.3	启动Tomcat.....	错误！未定义书签。
14.2.4	测试安装是否成功.....	错误！未定义书签。
14.3	部署一个Web应用.....	错误！未定义书签。
14.3.1	以文件夹方式部署一个Web应用.....	错误！未定义书签。
14.3.2	以WAR方式布署Web应用.....	错误！未定义书签。
14.4	在Web页面配置Tomcat.....	错误！未定义书签。
14.4.1	用户与角色管理.....	错误！未定义书签。
14.4.2	配置server.xml文件.....	错误！未定义书签。
14.4.3	配置<context>组件.....	错误！未定义书签。
14.4.4	配置数据源.....	错误！未定义书签。
14.5	使用Servlet过滤器Filter.....	错误！未定义书签。
14.5.1	编写JSP文件.....	错误！未定义书签。
14.5.2	编写Servlet过滤器.....	错误！未定义书签。
14.5.3	编写配置文件.....	错误！未定义书签。
14.5.4	查看过滤器效果.....	错误！未定义书签。
14.5.5	过滤器的运行过程.....	错误！未定义书签。
14.6	Tomcat与Hibernate的结合使用.....	错误！未定义书签。
14.6.1	资源层的设计.....	错误！未定义书签。
14.6.2	Model层的设计.....	错误！未定义书签。
14.6.3	Hibernate实现的DAO层.....	错误！未定义书签。
14.6.4	业务逻辑层（BM）的设计.....	错误！未定义书签。
14.6.5	展现层JSP的设计.....	错误！未定义书签。
14.7	小结.....	错误！未定义书签。
第十五章	Ant基本应用及Hibernate整合.....	错误！未定义书签。
15.1	Ant简介.....	错误！未定义书签。
15.2	Ant的下载及安装.....	错误！未定义书签。
15.2.1	Ant的下载.....	错误！未定义书签。
15.2.2	Ant的安装.....	错误！未定义书签。
15.2.3	使用ant的一个例子.....	错误！未定义书签。
15.3	建立工程目录.....	错误！未定义书签。
15.4	构建文件.....	错误！未定义书签。
15.4.1	project元素.....	错误！未定义书签。
15.4.2	target元素.....	错误！未定义书签。
15.4.3	task.....	错误！未定义书签。
15.5	Ant数据元素.....	错误！未定义书签。
15.6	DataType.....	错误！未定义书签。
15.6.1	Environment环境变量.....	错误！未定义书签。
15.6.2	argument参数.....	错误！未定义书签。
15.6.3	fileset文件集.....	错误！未定义书签。

15.6.4	patternset模式集 .....	错误！未定义书签。
15.6.5	path路径 .....	错误！未定义书签。
15.6	Ant核心任务 .....	错误！未定义书签。
15.6.1	<copy> .....	错误！未定义书签。
15.6.2	<delete> .....	错误！未定义书签。
15.6.3	<mkdir> .....	错误！未定义书签。
15.6.4	<javac> .....	错误！未定义书签。
15.6.5	<java> .....	错误！未定义书签。
15.6.6	<jar> .....	错误！未定义书签。
15.7	Ant自定义任务 .....	错误！未定义书签。
15.7.1	扩展org.apache.tools.ant.Task类 .....	错误！未定义书签。
15.7.2	编写<taskdef>标签 .....	错误！未定义书签。
15.7.3	使用自定义标签 .....	错误！未定义书签。
15.8	Ant与Hibernate的结合使用 .....	错误！未定义书签。
15.8.1	编写build.xml文件 .....	错误！未定义书签。
15.8.2	运行程序 .....	错误！未定义书签。
15.9	小结 .....	错误！未定义书签。
第十六章	JUnit基本应用 .....	错误！未定义书签。
16.1	JUnit简介 .....	错误！未定义书签。
16.1.1	什么是单元测试 .....	错误！未定义书签。
16.1.2	为何要使用JUnit .....	错误！未定义书签。
16.1.3	JUnit下载及安装配置 .....	错误！未定义书签。
16.2	JUnit基本应用 .....	错误！未定义书签。
16.2.1	不使用JUnit的测试例子 .....	错误！未定义书签。
16.2.2	第一个使用JUnit的简单测试例子 .....	错误！未定义书签。
16.2.3	用fixture来管理资源 .....	错误！未定义书签。
16.2.4	TestSuite .....	错误！未定义书签。
16.3	使用Mock进行测试 .....	错误！未定义书签。
16.3.1	使用JMock进行测试 .....	错误！未定义书签。
16.3.2	使用EasyMock进行测试 .....	错误！未定义书签。
16.3.3	测试数据库应用程序 .....	错误！未定义书签。
16.4	Cactus测试 .....	错误！未定义书签。
16.4.1	常规的Servlet容器内测试 .....	错误！未定义书签。
16.4.2	使用Cactus测试Servlet .....	错误！未定义书签。
16.4.3	使用Cactus测试JSP .....	错误！未定义书签。
16.5	Ant与JUnit的结合使用 .....	错误！未定义书签。
16.5.1	自动建构与测试 .....	错误！未定义书签。
16.5.2	自动生成测试报告 .....	错误！未定义书签。
16.6	JUnit与Hibernate的结合使用 .....	错误！未定义书签。
16.6.1	搭建Hibernate环境 .....	错误！未定义书签。
16.6.2	对DAO方法进行测试 .....	错误！未定义书签。
16.7	JUnit用法总结 .....	错误！未定义书签。
第十七章	Log4j的基本应用 .....	错误！未定义书签。
17.1	Log4j简介 .....	错误！未定义书签。

17.2	Log4j的基本概念 .....	错误！未定义书签。
17.2.1	Logger：日志记录器 .....	错误！未定义书签。
17.2.2	Appender：输出端 .....	错误！未定义书签。
17.2.3	Layout：日志格式化器 .....	错误！未定义书签。
17.3	Log4j的配置 .....	错误！未定义书签。
17.3.1	默认的Log4j初始化过程 .....	错误！未定义书签。
17.3.2	BasicConfigurator.configure()方法 .....	错误！未定义书签。
17.3.3	PropertyConfigurator.configure()方法 .....	错误！未定义书签。
17.3.4	DOMConfigurator.configure()方法 .....	错误！未定义书签。
17.3.5	定义多个输出目的地的实例 .....	错误！未定义书签。
17.4	小结 .....	错误！未定义书签。
第十八章	Struts入门及Hibernate整合 .....	错误！未定义书签。
18.1	Struts及MVC简介 .....	错误！未定义书签。
18.1.1	什么是MVC模式 .....	错误！未定义书签。
18.1.2	Struts实现的MVC设计模式介绍 .....	错误！未定义书签。
18.2	开发第一个Struts应用 .....	错误！未定义书签。
18.2.1	拷贝Struts需要的文件到\WEB-INF\lib\目录下 .....	错误！未定义书签。
18.2.2	编写ActionForm和Action的子类 .....	错误！未定义书签。
18.2.3	编写配置文件 .....	错误！未定义书签。
18.2.4	编写JSP文件 .....	错误！未定义书签。
18.3	消息资源的配置 .....	错误！未定义书签。
18.3.1	创建资源包 .....	错误！未定义书签。
18.3.2	指定资源包 .....	错误！未定义书签。
18.3.3	资源文件的存放位置 .....	错误！未定义书签。
18.3.4	引用资源文件中的信息 .....	错误！未定义书签。
18.4	使用ActionForm .....	错误！未定义书签。
18.4.1	配置ActionForm .....	错误！未定义书签。
18.4.2	validate()验证 .....	错误！未定义书签。
18.5	使用Action .....	错误！未定义书签。
18.5.1	<action>标签 .....	错误！未定义书签。
18.5.2	<forward>标签 .....	错误！未定义书签。
18.6	使用DynaActionForm .....	错误！未定义书签。
18.7	使用DispatchAction .....	错误！未定义书签。
18.8	Struts标签库 .....	错误！未定义书签。
18.9	使用bean标签库 .....	错误！未定义书签。
18.9.1	<bean:write>标签 .....	错误！未定义书签。
18.9.2	<bean:message>标签 .....	错误！未定义书签。
18.10	HTML标签 .....	错误！未定义书签。
18.10.1	<html:errors>标签 .....	错误！未定义书签。
18.10.2	<html:messages>标签 .....	错误！未定义书签。
18.11	Logic标签 .....	错误！未定义书签。
18.11.1	条件逻辑 .....	错误！未定义书签。
18.11.2	遍历标签 .....	错误！未定义书签。
18.11.3	转发和重定向标签 .....	错误！未定义书签。

18.12	Struts与Hibernate、MySql、Tomcat的整合使用 .....	1
18.12.1	搭建环境 .....	14
18.12.2	编写持久层代码 .....	14
18.12.3	编写业务层代码 .....	16
18.12.4	编写展现层 .....	18
18.12.5	其他的配置 .....	22
18.13	小结 .....	错误！未定义书签。
第十九章	Spring基本应用及与Hibernate的整合 .....	错误！未定义书签。
19.1	第一个HelloWord程序 .....	错误！未定义书签。
19.1.1	下载及配置Spring .....	错误！未定义书签。
19.1.2	编写一个Java文件 .....	错误！未定义书签。
19.1.3	编写Spring配置文件 .....	错误！未定义书签。
19.1.4	编写测试程序 .....	错误！未定义书签。
19.2	Constructor注入 .....	错误！未定义书签。
19.3	设置数据源 .....	错误！未定义书签。
19.3.1	编写测试数据源的程序 .....	错误！未定义书签。
19.3.2	编写bean.xml文档 .....	错误！未定义书签。
19.3.3	运行测试程序 .....	错误！未定义书签。
19.4	编程式事务管理 .....	错误！未定义书签。
19.4.1	编写实体类文件和Hibernate配置文件 .....	错误！未定义书签。
19.4.2	编写DAO .....	错误！未定义书签。
19.4.3	编写bean.xml文档 .....	错误！未定义书签。
19.4.4	运行DAO .....	错误！未定义书签。
19.5	声明式事务管理 .....	错误！未定义书签。
19.5.1	编写DAO .....	错误！未定义书签。
19.5.2	编写bean.xml文档 .....	错误！未定义书签。
19.5.3	运行DAO .....	错误！未定义书签。
19.6	Spring与Hibernate的结合使用 .....	错误！未定义书签。
19.6.1	编写Java实体类 .....	错误！未定义书签。
19.6.2	编写DAO .....	错误！未定义书签。
19.6.3	编写bean.xml文档 .....	错误！未定义书签。
19.6.4	运行测试程序 .....	错误！未定义书签。
19.7	小结 .....	错误！未定义书签。
第二十章	Hibernate常用工具的使用 .....	错误！未定义书签。
20.1	Middlegen-Hibernate（由ddl生成hbm） .....	错误！未定义书签。
20.1.1	建立数据表 .....	错误！未定义书签。
20.1.2	配置Middlegen-Hibernate .....	错误！未定义书签。
20.1.3	运行Middlegen-Hibernate .....	错误！未定义书签。
20.1.4	生成hbm文件 .....	错误！未定义书签。
20.2	Hibernate-extensions（由hbm生成java） .....	错误！未定义书签。
20.2.1	配置Hibernate-extensions .....	错误！未定义书签。
20.2.2	运行hbm2java.bat工具 .....	错误！未定义书签。
20.3	SchemaExport（由hbm生成ddl） .....	错误！未定义书签。
20.3.1	编写hbm文件 .....	错误！未定义书签。

20.3.2	使用SchemaExport从hbm生成数据库脚本ddl .....	错误！未定义书签。
20.3.3	在Ant中使用SchemaExport .....	错误！未定义书签。
20.4	Hibernate-extensions控制台 .....	错误！未定义书签。
20.4.1	配置Console .....	错误！未定义书签。
20.4.2	查看映射关系 .....	错误！未定义书签。
20.4.3	查询对象 .....	错误！未定义书签。
20.5	XDoclet ( 由java生成hbm ) .....	错误！未定义书签。
20.5.1	XDoclet在Hibernate中的用法 .....	错误！未定义书签。
20.5.2	编写java文件 .....	错误！未定义书签。
20.5.3	编写Ant的buildfile .....	错误！未定义书签。
20.6	P6SPY及SQL Profiler .....	错误！未定义书签。
20.6.1	配置P6SPY .....	错误！未定义书签。
20.6.2	查看P6SPY生成的日志 .....	错误！未定义书签。
20.6.3	配置SQL Profiler .....	错误！未定义书签。
20.6.4	查看SQL Profiler的信息 .....	错误！未定义书签。
20.7	使用HibernateSynchronize .....	错误！未定义书签。
20.7.1	下载和完装HibernateSynchronize .....	错误！未定义书签。
20.7.2	建立一张数据表 .....	错误！未定义书签。
20.7.3	生成hibernate.cfg.xml文件 .....	错误！未定义书签。
20.7.4	生成java类和XXX.hbm.xml文件 .....	错误！未定义书签。
20.8	小结 .....	错误！未定义书签。
第二十一章	Hibernate+Struts+Spring的结合应用 .....	错误！未定义书签。
21.1	生成数据表脚本 .....	错误！未定义书签。
21.1.1	编写Java实体类文件 .....	错误！未定义书签。
21.1.2	编写hibernate.cfg.xml .....	错误！未定义书签。
21.1.3	在Ant中运行Xdoclet与SchemeExport .....	错误！未定义书签。
21.2	DAO访问层 .....	错误！未定义书签。
21.2.1	编写接口 .....	错误！未定义书签。
21.2.2	实现接口 .....	错误！未定义书签。
21.3	业务逻辑层 .....	错误！未定义书签。
21.3.1	编写业务逻辑类 .....	错误！未定义书签。
21.3.1	编写Spring配置文件bean.xml .....	错误！未定义书签。
21.4	编写UI层 .....	错误！未定义书签。
21.4.1	编写web.xml文件 .....	错误！未定义书签。
21.4.2	编写Action与ActionForm .....	错误！未定义书签。
21.4.3	编写struts-config.xml .....	错误！未定义书签。
21.4.4	编写JSP页面 .....	错误！未定义书签。
21.5	查看运行效果 .....	错误！未定义书签。
21.6	小结 .....	错误！未定义书签。

## 18.12 Struts与Hibernate、 MySql、 Tomcat的整合使用

本小节介绍在 Tomcat 作为 Web 服务器 ,MySQL 作为数据库服务器的情况下 ,结合使用 Stuts 与 Hibernate , 分层结构如图 18-9 :



图18-9 系统分层结构

### 18.12.1 搭建环境

本节中使用到一个 Student 类 ,Student 类在 MySQL 对应的表是 student 表 ,建表语句如下 所示 :

```
//student表
CREATE TABLE student (
  id varchar(100) NOT NULL default '',
  name varchar(20) default '',
  `cardId` varchar(20) NOT NULL default '',
  age int(11) default '0',
  PRIMARY KEY (id)
) ENGINE = InnoDB
CHARACTER SET utf8 COLLATE utf8_general_ci;
```

### 18.12.2 编写持久层代码

student 表建立以后 ,就可以编写持久层代码并进行测试 ,查看持久层与资源层的结合使用是否正常。

#### 1. 编写实体类文件

学生类 Student.java 的部分代码如下所示。

```
//Student.java
public class Student {
  private String id; //标识id
  private String cardId; //学号
  private String name; //学生姓名
  private int age; //岁数
  //省略get和set方法
}
```

Student.java 实体所对应的配置文件为 :

```
//Student.hbm.xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping
  PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="model.Student" table="student" lazy="true"
    select-before-update="true"><!--把类和数表关联起来-->
    <id name="id" unsaved-value="null"><!--id的产生方式是uuid.hex-->
      <generator class="uuid.hex" />
    </id>
  </class>
</hibernate-mapping>
```

```

        </id>
        <property name="cardId" type="string" /><!--映射号-->
        <property name="name" type="string" /><!--映射学生名-->
        <property name="age" type="int" /><!--映射学生岁数-->
    </class>
</hibernate-mapping>

```

## 2. 编写Hibernate配置文件

Hibernate 的配置文件源代码如下：

```

//hibernate.cfg.xml
.....
<hibernate-configuration>
    <session-factory>
        <property name="show_sql">true</property><!--不显示SQL语言-->
        <property name="connection.driver_class"><!--指定连接MySQL的驱动-->
            com.mysql.jdbc.Driver
        </property>
        <property name="connection.url">

            jdbc:mysql://localhost:3306/schoolProject?useUnicode=true&characterEn
coding=GBK<!--连接数据库的URL，库名是schoolProject，
useUnicode=true&characterEncoding=GBK这句话的意思是采用中文编码-->
        </property>
        <property name="connection.username"><!--连接的登录名-->
            root
        </property>
        <property name="connection.password"><!--登录密码-->
            pwd
        </property>
        <property name="dialect"><!--指定连接的语言-->
            org.hibernate.dialect.MySQLDialect
        </property>
        <mapping resource="Student.hbm.xml" /><!--映射Student这个资源-->
    </session-factory>
</hibernate-configuration>

```

## 3. 编写Session的管理理

在 HibernateUtil.java 类中编写 SessionFactory 的获得，以及 Session 实例的打开和关闭代码，如下所示：

```

// HibernateUtil.java
public class HibernateUtil {
    private static final SessionFactory sessionFactory;
    //取得sessionFactory
    static {
        try {
            sessionFactory = new Configuration().configure()
                .buildSessionFactory();
        } catch (HibernateException ex) {
            throw new RuntimeException("Exception building SessionFactory: "
                + ex.getMessage(), ex);
        }
    }
    //开启session
    public static Session currentSession() {
        Session s = sessionFactory.openSession();
        return s;
    }
}

```

```

    }
    //关闭session
    public static void closeSession(Session s){
        s.close();
    }
}

```

### 18.12.3 编写业务层代码

在本小节的示例中，业务方法比较简单，包括对学生信息的增加、删除、修改、浏览共 4 个操作。如下是 StudentDAO.java 的代码：

```

//StudentDAO.java
public class StudentDAO extends BaseDAO {
    static Session session = null;
    static Transaction tx = null;
    static Logger logger = Logger.getLogger(StudentDAO.class);
    //根据id查找一名学生
    public static Student findById(String id) {
        Student stu = null;
        try {
            session = HibernateUtil.currentSession(); //开启连接
            tx = session.beginTransaction(); //开启事务
            stu = (Student) session.get(Student.class, id);
            tx.commit();
        } catch (HibernateException e) { //捕捉例外
            e.printStackTrace();
            tx.rollback();
        } finally {
            HibernateUtil.closeSession(session);
        }
        return stu;
    }
    //根据名字查找学生集合
    public static List findByName(String name) {
        List list = null;
        try {
            session = HibernateUtil.currentSession(); //开启连接
            tx = session.beginTransaction(); //开启事务
            Query q=session.createQuery("from Student s where s.name=:name");
            q.setString(1,name);
            list = q.list();
            tx.commit();
        } catch (HibernateException e) { //捕捉例外
            e.printStackTrace();
            tx.rollback();
        } finally {
            HibernateUtil.closeSession(session);
        }
        return list;
    }
    //查找所有学生的集合
    public static List getAllStu()
    {
        List list = null;
        try {
            session = HibernateUtil.currentSession(); //开启连接

```



```

        tx = session.beginTransaction();           //开启事务
        list = session.createQuery("from Student")
            .list();
        tx.commit();
    } catch (HibernateException e) {               //捕捉例外
        e.printStackTrace();
        tx.rollback();
    } finally {
        HibernateUtil.closeSession(session);
    }

    return list;
}
//测试函数，向student表插入一条记录
public static void main(String ars[]) {
    Student newStu=new Student();
    newStu.setName("tomclus");
    newStu.setCardId("12345");
    newStu.setAge(26);
    StudentDAO.createObj(newStu);
}
}

```

StudentDAO 继承自 BaseDAO 类，在 BaseDAO 类中封装了大部分常用的 DAO 操作，这种使用继承的方式比较利于程序扩展。下面是 BaseDAO.java 的代码：

```

//BaseDAO.java
public class BaseDAO {
    static Session session=null;
    static Transaction tx=null;
    /*-----创建新对象-----*/
    public static void createObj(Object o) {
        try {
            session = HibernateUtil.currentSession(); //开启连接
            tx = session.beginTransaction();           //开启事务
            session.save(o);
            tx.commit();
        } catch (HibernateException e) {             //捕捉例外
            e.printStackTrace();
            tx.rollback();
        } finally {
            {
                if(session!=null)
                    HibernateUtil.closeSession(session);
            }
        }
    }
    /*-----删除对象-----*/
    public static void delObject(String id) {
        try {
            session = HibernateUtil.currentSession(); //开启连接
            tx = session.beginTransaction();           //开启事务
            Object o=session.get(Student.class,id);
            session.delete(o);
            tx.commit();
        } catch (HibernateException e) {             //捕捉例外
            e.printStackTrace();
            tx.rollback();
        } finally
    }
}

```

```

        {HibernateUtil.closeSession(session);}
    }
    /*-----修改对象-----*/
    public static void mdfObj(Object o) {
        try {
            session = HibernateUtil.currentSession(); //开启连接
            tx = session.beginTransaction();           //开启事务
            session.update(o);
            tx.commit();
        } catch (HibernateException e) {              //捕捉例外
            e.printStackTrace();
            tx.rollback();
        } finally
        {HibernateUtil.closeSession(session);}
    }
}

```

#### 18.12.4 编写展现层

展现层的目的是提供友好的界面传递信息并与用户交互。JSP 只是负责输入输出，它没有处理数据的能力；而 Action 则把将要传给 JSP 页面的数据处理准备好，把数据放入 request 中，然后把数据导向输出的页面。

##### 1. 编写ActionForm

在持久层有一个 Student 类封装了持久数据的信息，它与展现层的 ActionForm 意义不一样。下面再来编写一个 StudentForm。

```

//StudentForm.java
public class StudentForm extends ActionForm {
    private String id;
    private String cardId;
    private String name;
    private int age;
    //省略get和set方法
}

```

在一般情况下，ActionForm 与 Student 类的代码基本是一样的，之所以要编写两个相似的类，是因为这样可以减少层之间的耦合性。如果展现层和持久层都使用 ActionForm 作数据载体的话，就把 Struts 和 Hibernate 绑定在了一起，假如项目变更，决定不使用 Struts 而用另外的展现层框架，那么对 Hibernate 持久层的改动也是相当大的。比如在 Struts 中使用 DynaActionForm 还是 ActionForm，对于持久层来说是无关紧要的，只需要在 Action 中把 DynaActionForm 或是 ActionForm 转为持久层对象 PO，然后传给持久层就行了

##### 2. 编写Action

Action 就好像是数据指路牌，处理数据并将数据导向正确的页面。下面是 StuAction.java 的代码：

```

//StuAction.java
public class StuAction extends DispatchAction {
    //添加学生记录
    public ActionForward insert(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        StudentForm stu_vo=(StudentForm)form;
        Student stu_po=new Student();
        BeanUtils.copyProperties(stu_po, stu_vo);
    }
}

```

```

        StudentDAO.createObj(stu_po);
        List list = StudentDAO.getAllStu();
        request.setAttribute("list", list);
        return list(mapping, form, request, response);
    }

    //删除学生记录
    public ActionForward del(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        StudentForm stuForm = (StudentForm) form;
        StudentDAO.delObject(stuForm.getId());
        return list(mapping, form, request, response);
    }

    //取得要修改的学生资料，并把页面导向detail.sjp
    public ActionForward getMdfInfo(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        StudentForm stu_vo = (StudentForm) form;
        Student stu_po = StudentDAO.findById(stu_vo.getId());
        BeanUtils.copyProperties(stu_vo, stu_po);
        request.setAttribute("stuForm", stu_vo);
        return mapping.findForward("detail");
    }

    //修改学生记录
    public ActionForward update(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        StudentForm stu_vo = (StudentForm) form;
        Student stu_po = new Student();
        BeanUtils.copyProperties(stu_po, stu_vo);
        StudentDAO.mdfObj(stu_po);
        return list(mapping, form, request, response);
    }

    //取得学生列表，并发送到stu_list.jsp。需要把PO对象转为VO对象
    public ActionForward list(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        List list = StudentDAO.getAllStu();
        List temp = new ArrayList(list.size());
        for (int i = 0; i < list.size(); i++) {
            Student stu_po = (Student) list.get(i);
            StudentForm stu_vo = new StudentForm();
            BeanUtils.copyProperties(stu_vo, stu_po);
            temp.add(stu_vo);
        }
        request.setAttribute("list", temp);
        return mapping.findForward("stu_list");
    }
}

```

在上述的代码中有 BeanUtils.copyProperties(stu\_po, stu\_vo)语句，它的作用是把 stu\_vo 的数据拷贝到 stu\_po 中，不然的话，要使用以下几条语句来完成相同的功能：

```

stu_po.setAge(stu_vo.getAge());
stu_po.setId(stu_vo.getId());

```

.....

### 3. struts-config.xml的配置

本例中用到了 ActionForm 和 Action , 需要在 struts-config.xml 对它们进行配置 , 如下所示 :

```
//struts-config.xml
.....
<struts-config>
  <form-beans>
    <form-bean name="stuForm" type="model.StudentForm" />
  </form-beans>
  <action-mappings>
    <action
      path="/StuAction"
      name="stuForm"
      type="action.StuAction"
      scope="request"
      input="/detail.jsp"
      parameter="method"
    >
      <forward name="stu_list" path="/stu_list.jsp" />
      <forward name="detail" path="/detail.jsp" />
    </action>
  </action-mappings>
  <message-resources parameter="resources.application"/>
</struts-config>
```

在上述的<action>标签中定义了 parameter="method" , 说明 Action 是 DispatchAction , 可以根据不同的 method 参数来调用不同的方法。<forward>标签指定了两个 JSP 资源 , 分别是 stu\_list.jsp 和 detail.jsp。

### 4. 编写stu\_list.jsp

stu\_list.jsp 用于显示所有学生的资料。stu\_list.jsp 的代码如下所示。

```
//stu_list.jsp
<%@ page
  language="java"
  import="java.util.*"
  import="model.StudentForm"
  contentType="text/html; charset=gb2312"
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<body>
<table border="1">
  <tr>
    <td>学号</td>
    <td>姓名</td>
    <td>年龄</td>
    <td>操作</td>
  </tr>
  <%
    //request的"list"属性在action中就被设置好了,在JSP页面只要取出来用就行
    List list=(List)request.getAttribute("list");
    for(int i=0;i<list.size();i++)
    {
      StudentForm stu=(StudentForm)list.get(i);
    }
  <%
  >
  <tr>
```

```
 <%=stu.getCardId()%></td>  <%=stu.getName()%></td>  <%=stu.getAge()%></td>  <a href="StuAction.do?id=<%=stu.getId()%>&method=del">删除</a> <a href="StuAction.do?id=<%=stu.getId()%>&method=getMdfInfo">修改</a></td> </tr> <%}%> </table> <a href="detail.jsp?method=insert">添加</a> </body> </html> | | | |
```

部署 src\18character\18.12\WebRoot\ 文件夹为 Web 应用 /18.12，打开 MySQL 把 src\18character\18.12\18.12.sql 脚本导入 schoolproject 数据库，在浏览器中输入 http://127.0.0.1:8080/18.12/StuAction.do?method=list，得到如图 18-10 所示的界面。

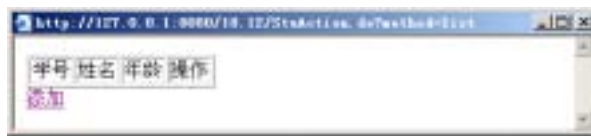


图18-10 用户信息列表

## 5. 编写detail.jsp

detail.jsp 有两个用途：

注册用户；

修改用户。

当 detail.jsp 在提交表单时，如果是注册用户，则 method 为 “insert”；如果是修改用户，则 method= “update”。

```

//detail.jsp
<%@ page
    language="java"
    contentType="text/html; charset=gb2312"
    import="model.StudentForm"
%>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<body>
<html:errors/>
<%
    //如果是修改用户，则从request流中取出将被修改的学生资料，提交时method="update"
    String method=request.getParameter("method");
    if(method.equals("getMdfInfo"))
    {
        StudentForm stuForm=(StudentForm)request.getAttribute("stuForm");
    }
%>
<form action="StuAction.do" method="post" >
    学号：<input type="text" name="cardId"
value="<%=stuForm.getCardId()%>"><br>
    姓名：<input type="text" name="name"
value="<%=stuForm.getName()%>"><br>
    年龄：<input type="text" name="age"
value="<%=stuForm.getAge()%>"><br>
    <input type="hidden" name="method" value="update">

```

```

        <input type="hidden" name="id" value="<%=stuForm.getId()%>">
        <input type="submit" value="提交">
    </form>
    //如果是注册用户，则表单中没有初始值，并且提交时method="insert"
    <%>else{<%>
        <form action="StuAction.do" method="post" >
        学号:<input type="text" name="cardId" ><br>
        姓名:<input type="text" name="name" ><br>
        年龄:<input type="text" name="age" ><br>
        <input type="hidden" name="method" value="insert">
        <input type="submit" value="提交">
    </form>
    <%><%>
</body>
</html>

```

在图中单击“添加”超链接，页面将转到 detail.jsp，如图 18-11 所示。

图18-11 添加用户界面

在图中填写资料后单击“提交”按钮，显示如图 18-12：

学号	姓名	年龄	操作
123456	tomclius	35	删除 修改

图18-12 用户信息列表

单击图中的“修改”超链接，将得到如图 18-13 所示画面。

图18-13 修改用户信息界面

在图中填写修改过后的信息后单击“提交”按钮，页面将转到 stu\_list.jsp 页面，并显示新的资料。

### 18.12.5 其他的配置

#### 1. web.xml

在 web.xml 中需要指定 ActionServlet 的位置，如下所示：

```

<web-app>
    <!-- Standard Action Servlet Configuration -->

```

```

<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>
    org.apache.struts.action.ActionServlet
  </servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
<!-- Standard Action Servlet Mapping -->
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
<taglib>
  <taglib-uri>/tags/struts-html</taglib-uri>
  <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
</taglib>
</web-app>

```

## 2. 配置tomcat数据源

在实际的应用中，使用数据源是很有必要的。本节的例子没有数据源，是为了方便讲解。在 Tomcat 中配置数据源很方便，配置好以后 hibernate.cfg.xml 中设置 hibernate.connection.datasource 属性值为数据源的 JNDI 名即可。