第 15 章　（课程）系统设计说明书样本

15.1　系统需求分析

15.1.1　功能需求

本管理系统需要满足监理公司在监理工程时，提供一个方便快捷的工程电子合同表格的管理平台。并提供了与之配套的人员管理，角色管理，和相关的权限管理。在后期还将完善统计功能。按照功能需求应该达到以下几点：实现监理公司以及与其有业务关系的建设单位，设计单位和各个施工单位人员的增删查改工作；实现监理公司按公司分类显示相关工程信息工作；实现监理公司所有业务合同信息的增删查改工作；实现监理公司以及与其有业务关系的建设单位，设计单位和各个施工单位的所有业务合同按人员职位的权限显示工作；完成监理公司以及与其有业务关系的建设单位，设计单位和各个施工单位的所有业务合同内容按人员职位的权限修改工作；实现监理公司工程新建和工程人员配置工作；实现监理公司以及与其有业务关系的建设单位，设计单位和各个施工单位人员的个人信息显示工作。实现监理公司管理者信息发布工作。实现监理公司以及与其有业务关系的建设单位，设计单位和各个施工单位人员的信息阅读与职位申请工作。实现监理公司对个公司人员属性统计工作。

15.1.2　可行性分析

软件开发者面向企业开发出基于 Web 的管理信息系统对企业各方面信息进行全面的管理，在一定程度上满足了企业的现代化管理要求，具体体现在以下几个方面：

- 经济性

当前许多中小型企业普遍采用的都是使用人工管理方式（即纸质材料以及笔类批复等）来管理企业的各类信息，这样的管理方式既管理困难，又浪费员工的时间和成本，并且容易出现对企业信息的疏漏，不利于企业发展的情况。具体到监理公司的业务特点，对于监理公司来说业务最重要的就是业务合同的各部门审核与批复，即如何实现合同的流程控制以及统计，在以往的工作过程中基本上采取就是人工管理方式，既费时又低效。因此开发出此 Web 系统，通过借鉴大型企业的先进管理方式，提高企业的工作效率，降低企业的运营成本。

- 技术性

正如上面所介绍的，在以往的企业业务合同流程以及权限管理过程中，采取的是传统的人工管理模式，容易被人钻漏洞，给公司造成一定的损失；而采取了 Web 管理系统之后，可以从技术层次杜绝这种现象，使公司各个等级的人员都必须按照规定的合同流程操作，消除了越级操作的可能性。

- 交互性

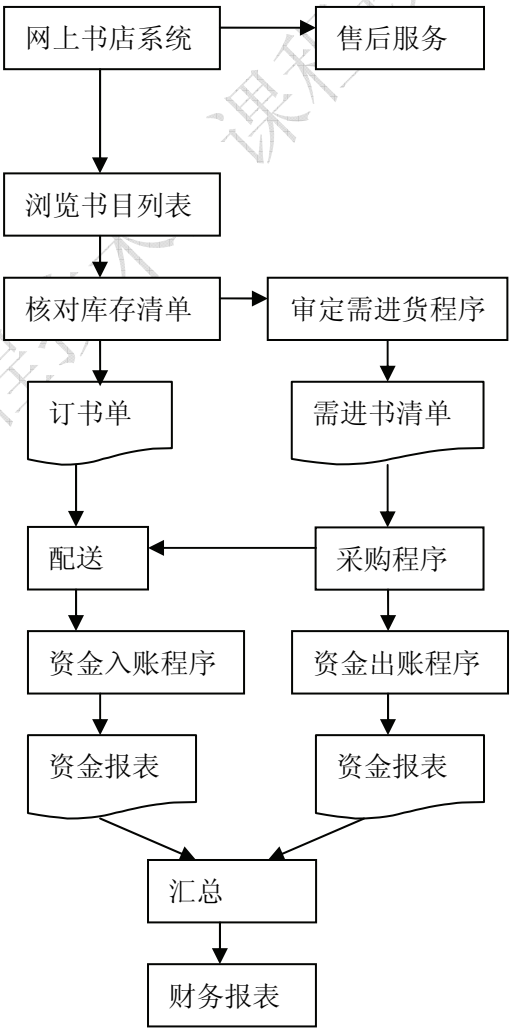
通过 Web 管理系统可以同时对企业日常办公进行管理，实现了企业员工之间的消息发送与接受功能，大大方便了企业内部人员的沟通，同时系统提供的公告栏管理，方便各级员工及时了解公司的动态以及相关信息，与系统的其他模块相互协助促进企业的管理进入科学系统的管理轨道。

实例分析：网上书店管理系统描述

系统功能描述

网上书店的工作流程分析:网上书店总共分四大部分，包括：用户管理，图书管理，销售管理，仓库管理，配送管理，售后管理。最重要的部分是销售管理他五个部分是以销售为中心的。用户管理包括已注册用户和未注册用户（未注册的先免费注册）。销售管理包括用户浏览图书列表，选择自己需要的图书，该图书要有详细介绍，包括作者，价格，内容简介，出版日期，出版社信息，ISBN 号等（这是图书管理所包括的内容），确认无误然后提交订单。顾客提交的订单直接送到进货部门处理，经核实信息准确有效后，交给仓库管理员配货，进货，打包，然后通知配送部门。在配送过程中顾客可以实时看到自己的订单的处理情况，比如正在配货中或正在配送途中，预计送达时间。在进货部门处理过程中要注意，要将订单发给离送货地最近的仓库，这样配送起来可以节约运输成本。配送部门将书送到顾客面前当面验货，满意后付款，配送部门要将书款交到财务并汇报订单已经完成。售后部门负责顾客退换货及投诉建议。

系统流程图



数据流图

表 1. 组成数据流图的元素可以从描述问题的信息中提取

源点/终点	处理（任何改变数据的操作）
用户 管理员	处理用户信息表 产生订单 处理图书信息表 处理事务
数据流	数据存储
用户信息表 　用户编号 　用户名 　密码 　联系方式 　送货地址 　邮箱 订单报表 　用户编号 图书编号 订购时间 价格 购买数量 图书信息表 图书编号 图书名称 作者 ISBN 号 价格 出版社名称 版次 库存量 事务（图书出库或入库） 　图书编号 　事务类型 　数量	用户信息 （见用户信息表） 订单信息 （见订单报表） 图书信息 （图书信息表） 库存清单 　图书编号 　库存量 　库存量临界值

为了方便阅读，我们将图书管理系统数据流图分成三层进行描绘。第一层是系统总概貌数据流图，第二层是销售管理数据流图，第三层是用户管理数据流图。

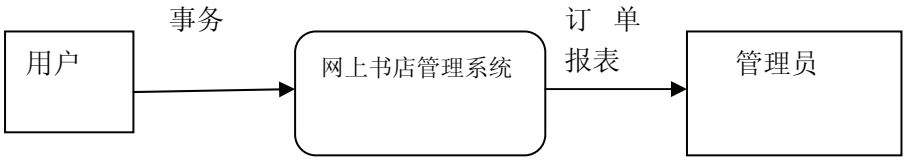


图 1. 1 网上书店管理系统基本模

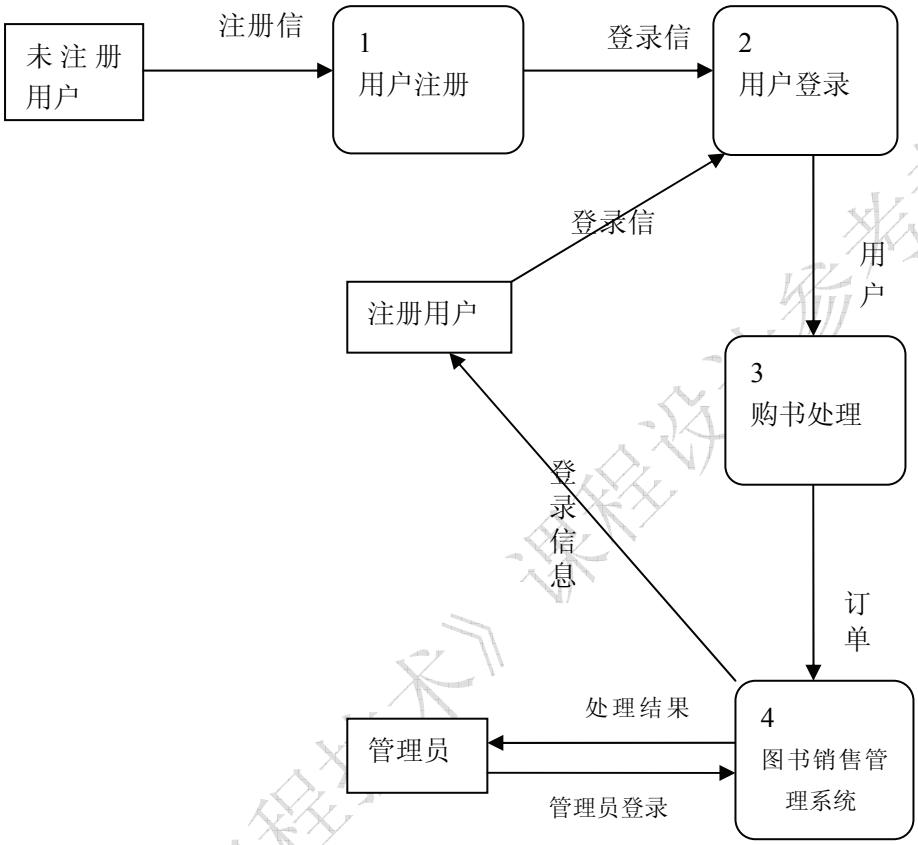
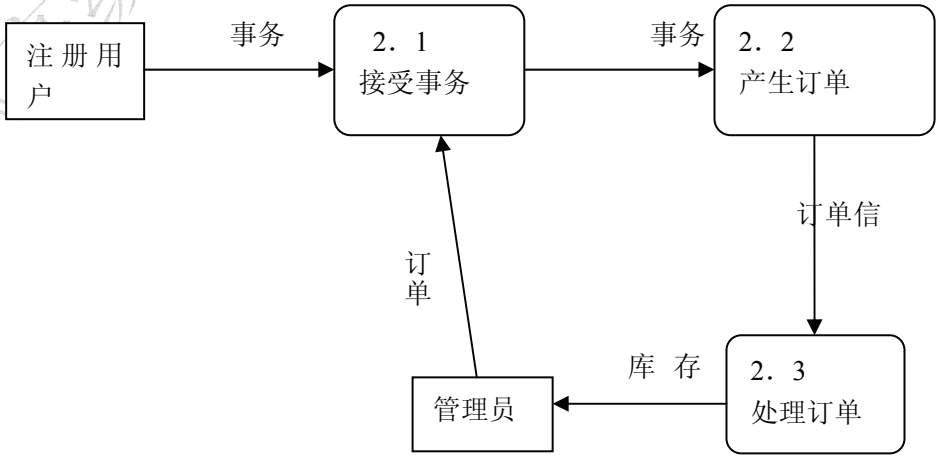
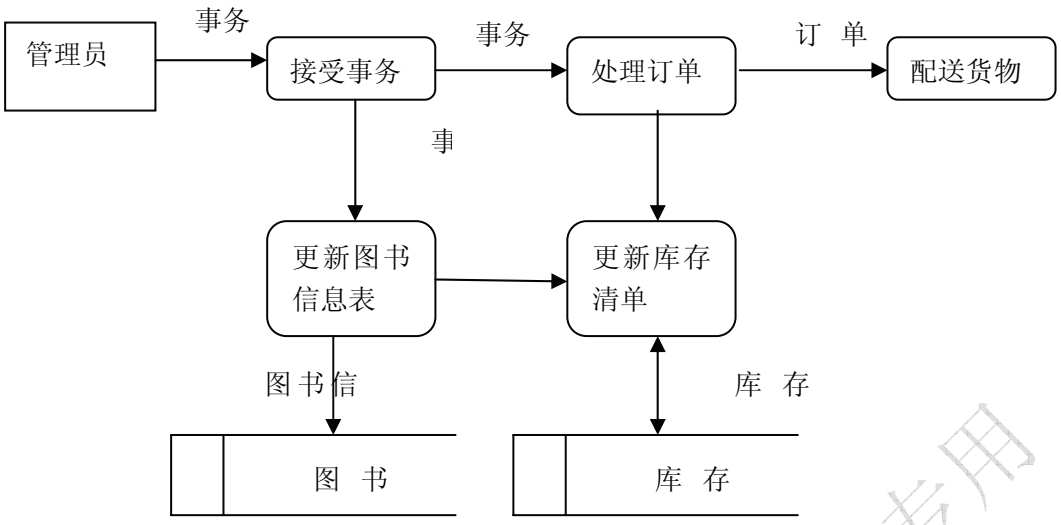


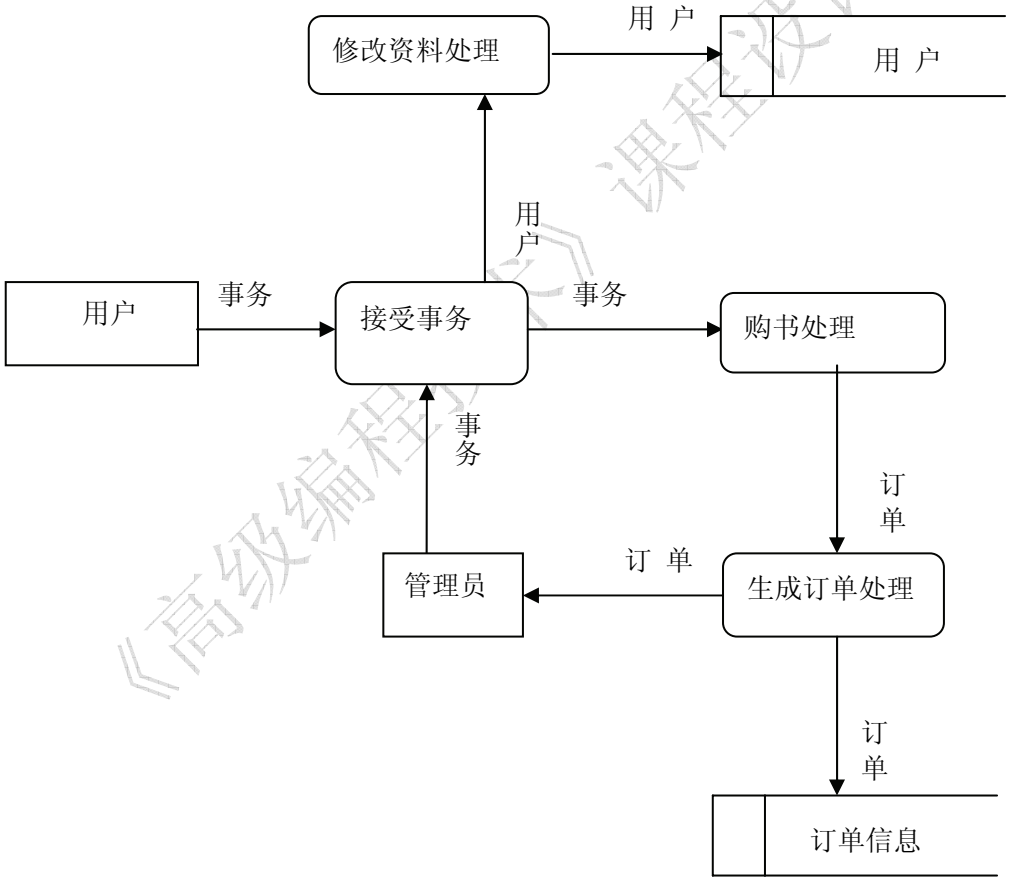
图 1.2 第一层系统总数据流图



第二层销售管理数据流图（前台管理）



第二层销售管理数据流图（后台管理）



第三层用户管理数据流图

根据数据流图生成的数据字典

名字：用户
别名：
描述：上网购书的人
定义：用户=[注册用户]+[非注册用户]
注册用户=用户编号+用户名+密码+联系方式+送货地址+邮箱
位置：用户信息表
订单报表
订单信息

名字：管理员
别名：
描述：更新，操作，维护网上书店管理系统的人
定义：
位置：

名字：接受事务
别名：
描述：接受外部输入
定义：接受事实=[接受顾客的输入]+[接受管理员的输入]
位置：用户信息表
订单信息表
图书信息表
库存清单
事务

名字：购书处理
别名：
描述：对用户所挑选的图书生成的订单的处理
定义：购书处理=当前清单+[最终清单]
位置：订单信息表
用户信息表
库存清单
事务

名字：修改资料处理
别名：
描述：用户资料的更新
定义：修改资料处理=[更新用户编号]+[更新用户名]+[更新密码]+[更新联系方式]+[更新送货地址]+[更新邮箱]
位置：用户信息表
用户信息

名字：生成订单处理
别名：
描述：用户订单的最终处理
定义：生成订单处理=订单编号+用户名+送货地址+图书名称+数量
位置：订单报表
订单信息

名字：用户信息表
别名：用户信息
描述：储存在该网注册过的用户的详细信息的表
定义：用户信息表=用户编号+用户名+密码+联系方式+送货地址+邮箱
位置：网上书店管理系统数据库

名字：订单报表
别名：订单信息
描述：储存该网购书订单的信息表
定义：订单报表=用户编号+图书编号+订购时间+价格+购买数量
位置：网上书店管理系统数据库
输出到打印机

名字：图书信息表
别名：图书信息
描述：储存该网销售的所有图书信息的表
定义：图书信息表=图书编号+图书名称+作者+ISBN 号+价格+出版社名称+版次+库存量
位置：网上书店管理系统数据库
订单报表
订单信息
库存清单

名字：库存清单
别名：
描述：记录仓库中所以图书数量的清单，以便管理员即使补充库存不足的图书。
定义：库存清单=图书编号+库存量+库存量临界值
位置：网上书店管理系统数据库

名字：图书编号
别名：
描述：唯一的标识库存清单中一本特定图书的关键域
定义：图书编号=13{数字}13
位置：图书信息表
图书信息
订单报表
订单信息
库存清单
事务

名字：用户编号
别名：
描述：唯一的标识用户信息表中一个特定用户的关键域
定义：用户编号=8{数字}8
位置：用户信息表
用户信息
订单报表
订单信息
事务

名字：订单编号
别名：
描述：唯一的标识订单报表中一个特定订单的关键域
定义：订单编号=13{数字}13
位置：订单报表
订单信息
事务

名字：购买数量
别名：
描述：某本图书一次订货的数量
定义：购买数量=1{数字}10
位置：订单报表
订单信息

15.2　数据库设计

15.2.1　数据库设计图

根据用户的需求，我将数据库分为三大实体部分和两大联系集：

(1) 项目工程部分，主要记录项目的各种信息，如参与项目的公司名称，工程的起始结束时间等。

(2) 人员管理部分，主要用于记录员工的信息。

(3) 合同表格部分，该部分分为 36 个模块，每个模块代表一张合同表格。这些实体集是虚实体集，他们没有主键，他们的分辨符是项目工程的 project_id。这种设计是为了便于一个工程存在多个同一类型的合同而设计的。

(4) 项目工程配置联系集，该集合联系了项目工程和各公司员工。使得工程的人员配置得到了统一管理。

(5) 工程角色联系集，该集通过描述工程角色和他们相关联的表格来形成集合主体，这个联系集合是人员和表格权限管理的重要依据。

系统数据库的设计实现，对系统的应用具有重要的指导意义。本系统采用 MySQL 数据库系统，图 15.1 给出了系统数据库的 ER 图设计。

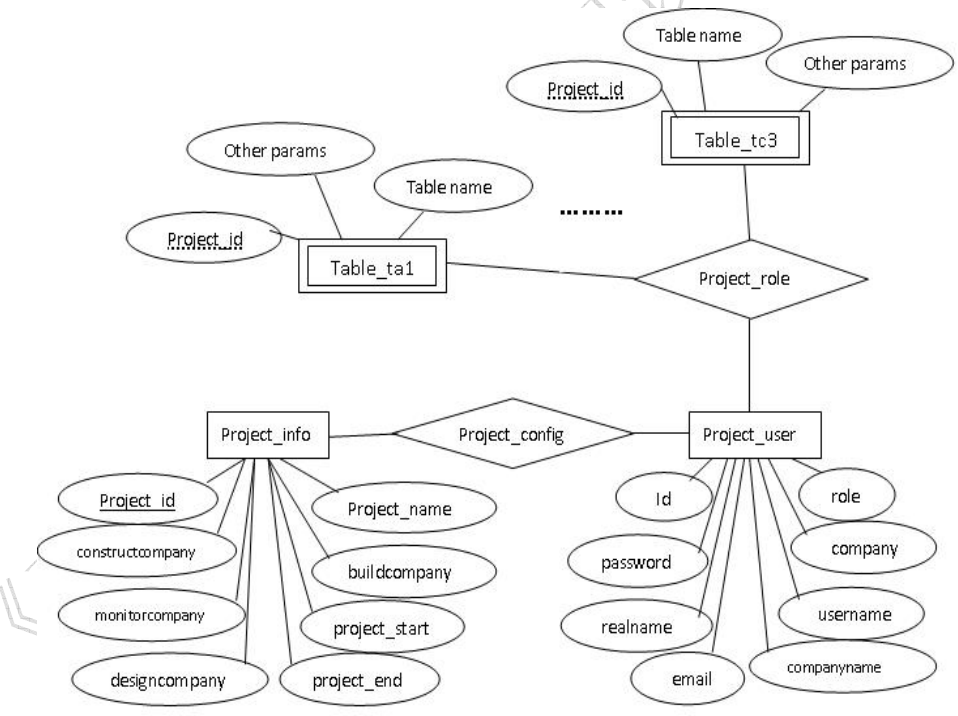


图 15.1　数据库 ER 设计

对 ER 图进行物理设计，并完成了与其相关的数据库结构设计，如图 15.2 系统数据库结构设计图：

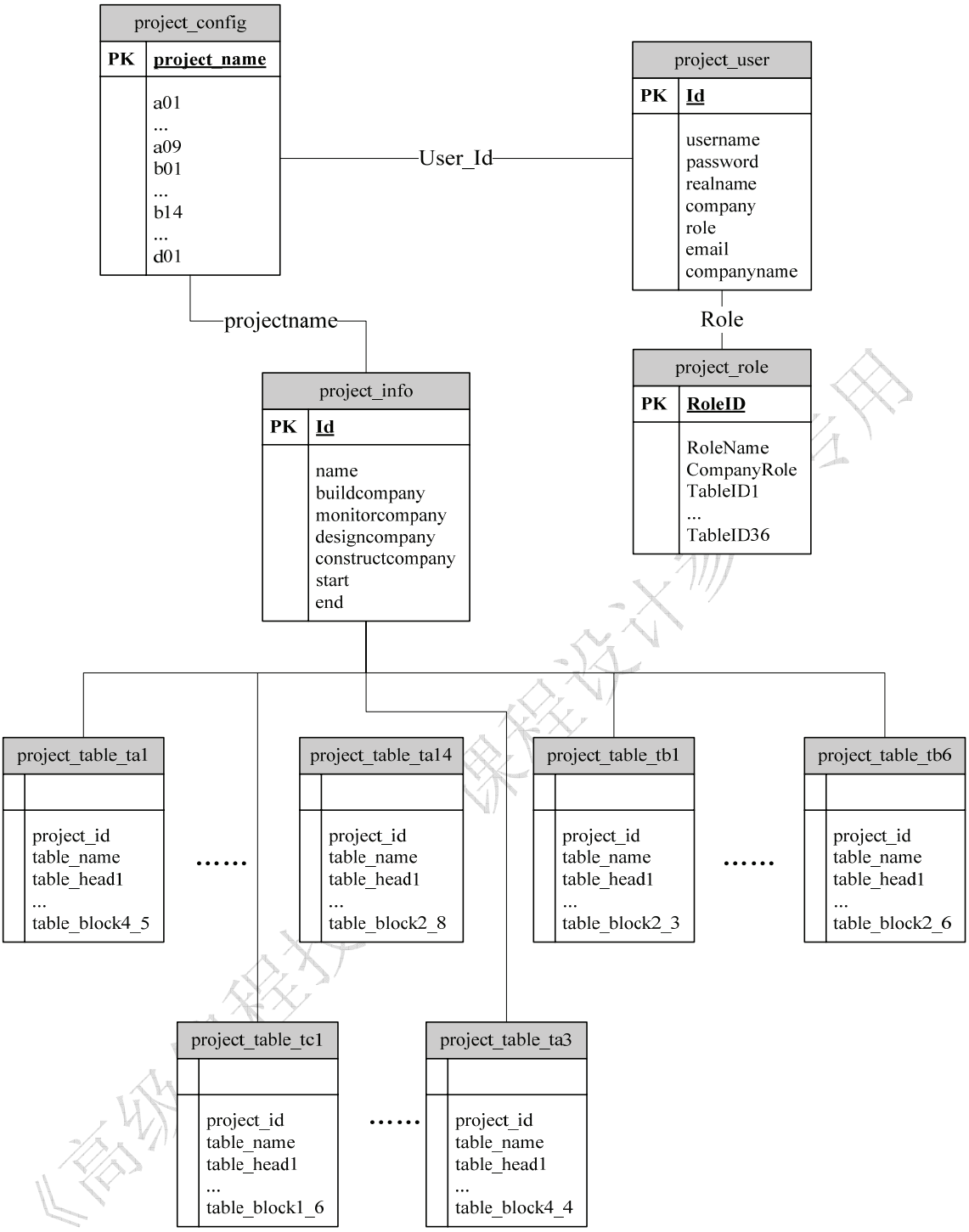


图 15.2　系统数据库结构设计图.

15.2.2　主要库表的结构

系统数据库命名为 projectdb，共包含 43 个表，由于系统相关数据表比较多，在此只列举出比较重要的数据表，其他数据表格式类似。

- project_user（用户信息表）

用户信息表的详细字段介绍如表 15.1 所示：

表 15.1　用户信息表.

字段名称	数据类型	长度	允许空	默认值	简介
Id	Int	11	否	无	用户 ID
Userame	Varchar	255	否	无	用户登录名
Password	Varchar	255	否	无	密码
Realname	Varchar	255	否	无	真实姓名
Company	Varchar	255	否	无	单位类别
Role	Varchar	255	是	无	职称/角色
Email	Varchar	255	否	无	邮箱地址
Companyname	Varchar	255	否	无	公司名称

其中 company 字段的值是实现权限管理关键，其具体的字段值值为建筑工程的四种单位类别：建设单位，监理单位，施工单位和设计单位。此表提供了各个单位人员的基本信息。

- project_config（工程配置表）

工程配置表的详细字段介绍如表 15.2 所示：

表 15.2　工程配置表.

字段名称	数据类型	长度	允许空	默认值	简介
Id	Int	11	否	无	工程 ID
Projectname	Varchar	255	否	无	工程名称
A01	Varchar	255	是	无	负责人
A02	Varchar	255	是	无	总经理
A03	Varchar	255	是	无	分管付总
A04	Varchar	255	是	无	主管副经理
A05	Varchar	255	是	无	总工
A06	Varchar	255	是	无	（工程）部长
A07	Varchar	255	是	无	工程部负责人
A08	Varchar	255	是	无	主管工程师/主管
A09	Varchar	255	是	无	项目负责人
B01	Varchar	255	是	无	负责人
B02	Varchar	255	是	无	项目经理
B03	Varchar	255	是	无	（施工）技术负责人
B04	Varchar	255	是	无	复核人/审核人
B05	Varchar	255	是	无	土建专业负责人
B06	Varchar	255	是	无	电气专业负责人
B07	Varchar	255	是	无	水暖专业负责人
B08	Varchar	255	是	无	质检员
B09	Varchar	255	是	无	自检员
B10	Varchar	255	是	无	专检员
B11	Varchar	255	是	无	填表人/编制人
B12	Varchar	255	是	无	工长
B13	Varchar	255	是	无	搅拌站
B14	Varchar	255	是	无	测量人
C01	Varchar	255	是	无	负责人
C02	Varchar	255	是	无	总项目管理工程师
C03	Varchar	255	是	无	专业项目管理工程师

C04	Varchar	255	是	无	项目管理工程师
C05	Varchar	255	是	无	总监
C06	Varchar	255	是	无	土建项目管理工程师
C07	Varchar	255	是	无	电气项目管理工程师
C08	Varchar	255	是	无	管道项目管理工程师
C09	Varchar	255	是	无	旁站项目管理人员
D01	Varchar	255	是	无	负责人

此表是系统人员和权限管理的关键，通过此表把与某工程相关的人员聚集在一起，方便管理员的配置和查阅，也为之后的权限管理打下了基础。

15.3　系统概要设计

15.3.1　系统开发及运行环境

- 硬件平台

（1）CPU：P4 1.73 GHz；
（2）内存：256MB 以上。

- 软件平台

（1）操作系统：Windows XP；
（2）数据库：MySQL 6.0.10；
（3）开发工具包：JDK Version 1.6.0；
（4）JSP 服务器：Tomcat 6.0；
（5）Java IDE：MyEclipse 6.5；
（6）浏览器：IE5.0 及以上版本，推荐 IE6.0；
（7）显示器分辨率：最佳效果 1024*768 像素。
（8）其他工具： Sitemesh、StrutsIDE1.2、Hibernate Synchronizer3.2 等插件。

15.3.2　系统用例分析

系统的操作类型分为四种：普通用户、管理员、管理系统和信息数据库。他们执行的用例分别是：接收信息、职位申请、请求合同列表、操作合同、权限判定、收到角色分配、工程人员配置、新建工程项目、更新信息和发布信息。如系统用例图 15.3 图所示：

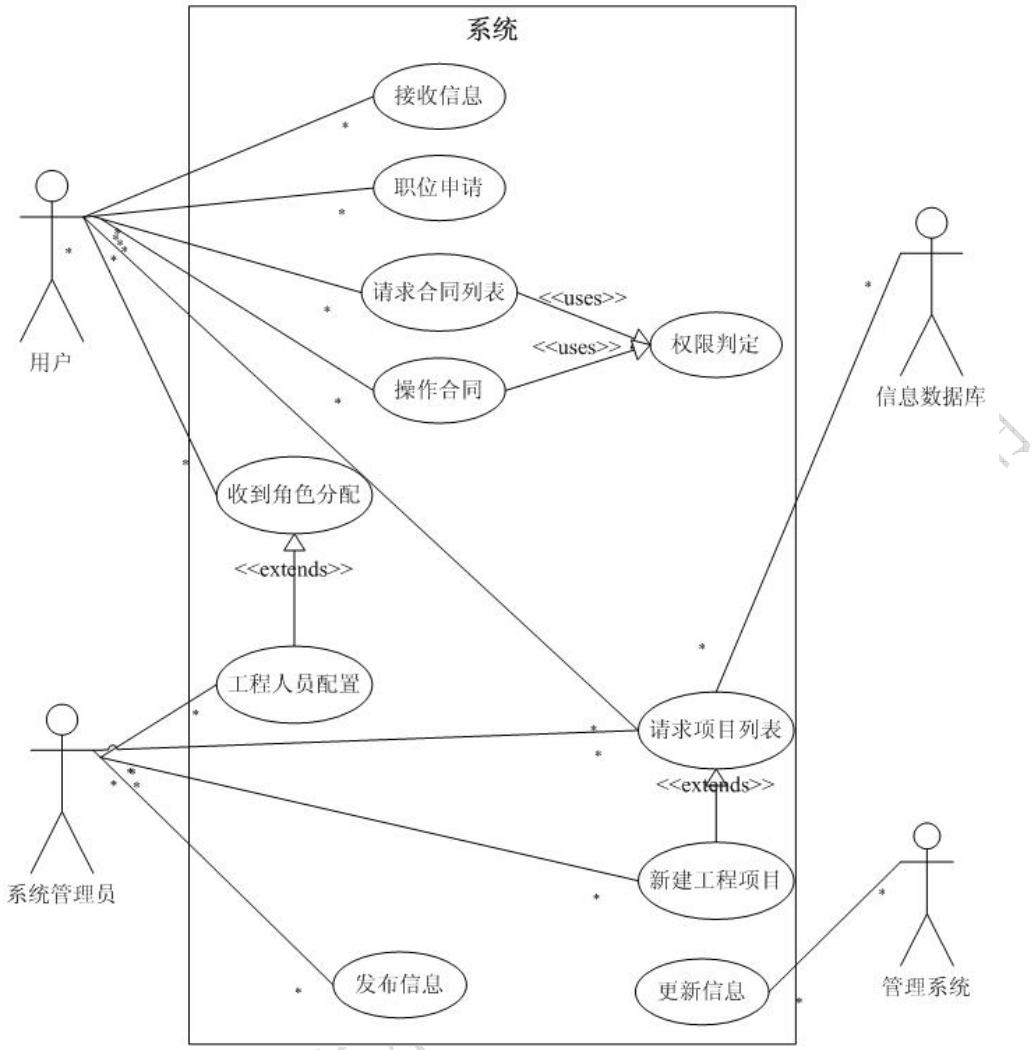


图 15.3　系统用例图

15.3.3　系统模块划分

系统主要由五个模块组成，它们分别是：用户管理、合同处理中心、项目管理、辅助办公、公共信息管理模块。

系统的功能结构图如图 15.4 所示。

图 15.4　系统功能结构图

15.3.4　系统模块简介

● 合同处理事务中心

此模块为系统的核心模块，主要负责公司合同相关信息的流程管理，合同流程图如图 15.5 所示：

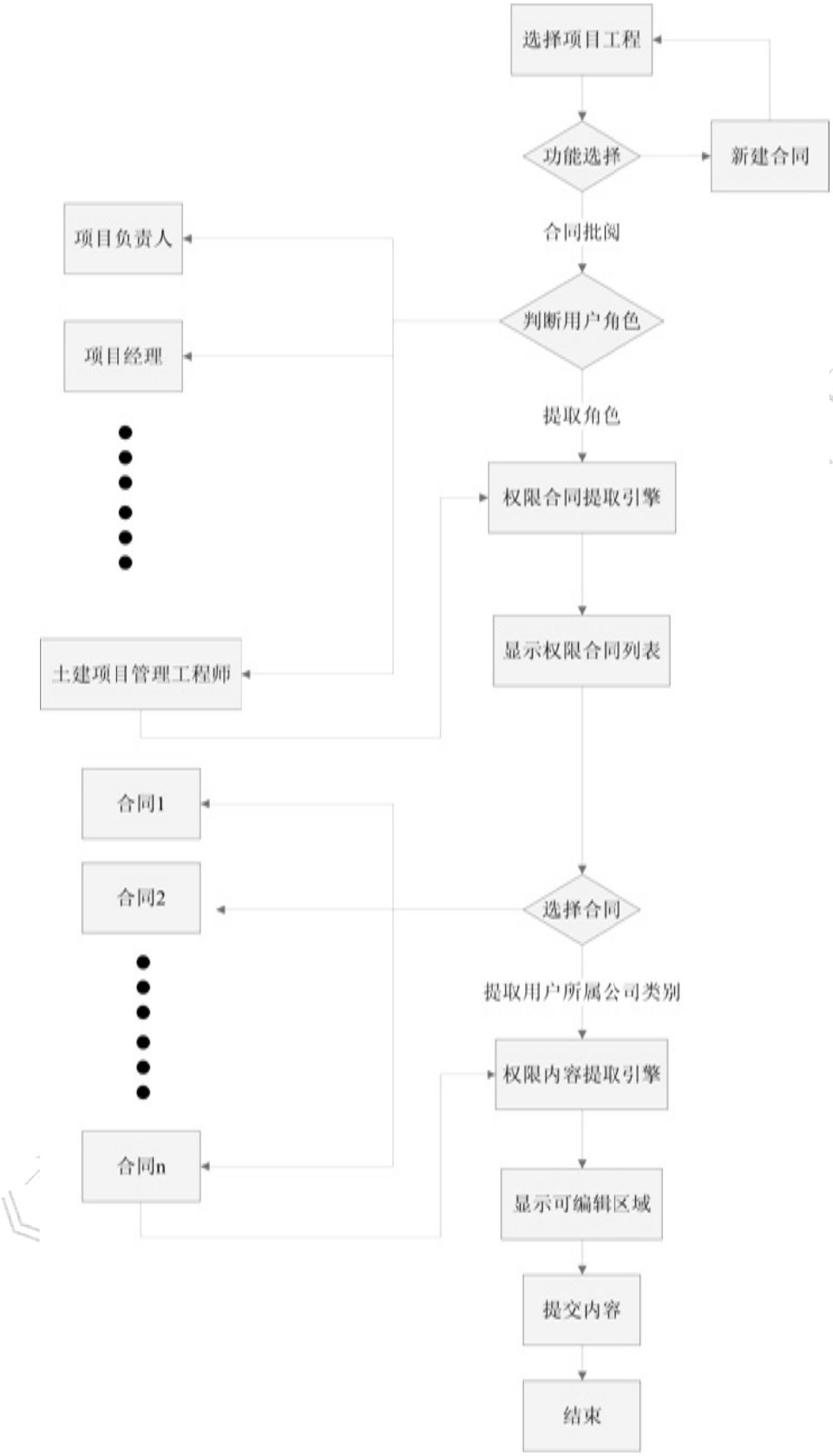


图 15.3　合同业务流程图

合同业务详细流程介绍如下：

- 进入所选项目，显示批阅和新建两种模式连接。

- 新建合同表格，讲为工程新加一个合同表格。
- 批阅合同时，权限引擎将会根据用户的角色身份来匹配与其相关的合同表格，并提供批阅接口。
- 对于显示的待批阅文件，用户可以进行批阅和编辑。
- 在进入待批阅文件中后，点击编辑按钮，系统根据用户的所属公司类型提取用户可以编辑的区域。
- 用户编辑完后，点击提交，将数据入档。

● 系统用户管理

本模块主要是提供对系统用户的注册查询等功能，实现对系统用户（即公司职员）的权限分配以及功能划分。由于公司业务合同是公司的重要财产，设计到了很多公司敏感数据的保护问题，因此系统的安全性和权限控制就显得尤为重要。本系统根据公司日常管理特点，对用户权限进行了较为详细的划分。

(1) 系统管理员角色：管理员。
(2) 建筑单位 9 个角色：负责人 ，总经理，分管付总主管副经理，总工，（工程）部长，工程部负责人，主管工程师/主管，项目负责人。
(3) 施工单位 14 个角色：负责人，项目经理，（施工）技术负责人，复核人/审核人，土建专业负责人，电气专业负责人，水暖专业负责人，质检员，自检员，专检员，填表人/编制人，工长，搅拌站，测量人。
(4) 监理单位 9 个角色：负责人，总项目管理工程师，专业项目管理工程师，项目管理工程师，总监，土建项目管理工程师，电气项目管理工程师，管道项目管理工程师，旁站项目管理人员。
(5) 设计单位 1 个角色：负责人。

● 系统项目管理

该模块主要负责监理公司对项目的管理，功能上分为：项目新建和项目配置。其中项目新建，包括项目名，项目的相关单位等基本信息。项目配置包括提取各个单位的角色信息，和与之匹配的人员名称，并实现项目与人员匹配的功能。

本模块只有管理员身份才能进入。

● 公共信息管理

该模块主要提供公司内部员工之间的信息交互，并且提供了公告栏，便于公司内部信息的及时沟通。其中职位申请部分主要实现了员工与管理者的交互，例如申请加入 XX 新项目工程的 XX 职位。公告栏则为公司领导发布各种信息提供了一个平台。此部分所有用户均可操作。

● 辅助办公

本模块主要提供了给管理人员对各个公司的人员情况进行统计的功能。用户在注册时，需填写职称等级一项。该模块对各个公司的人员进行统计，为管理人员的决策提供依据。

15.4　系统表示层实现

系统的表示层主要使用 Struts 框架的视图组件来实现。而 Struts 框架的视图负责为客户提供动态网页内容，其主要由 JSP 网页构成，此外，Struts 框架还提供了 Struts 客户化标签和 ActionForm Bean，这些组件对国际化、接受用户的表单数据、表单 Validator 验证和 Error 错误处理等的支持，使开发者把更多的精力放在实现业务需求上。接下来就具体介绍表示层的几种实现在系统中的应用。

15.4.1　Struts 视图层实现

视图部分是用户和系统的交互界面，是系统模型的外在表现形式，用户通过视图来了解模型的外在表现形式。要介绍 Struts 视图层 ActionForm Bean 的运行原理，必须首先介绍 JavaBean 的概念，从而更好地理解 ActionForm。JavaBean 是可重用的、平台独立的 Java 组件，它支持属性、事件、方法和持久化。而 Struts 框架仅利用了 JavaBean 的一部分特性。

ActionForm Bean 本身有两种存在范围：request 和 session。前者表示 ActionForm 实例及其包含的数据仅在当前的 Request/Response 生命周期中有效，而后者则表示在整个 HTTP Session 中均有效。AcitonForm 的生命周期如图 15.6 所示：

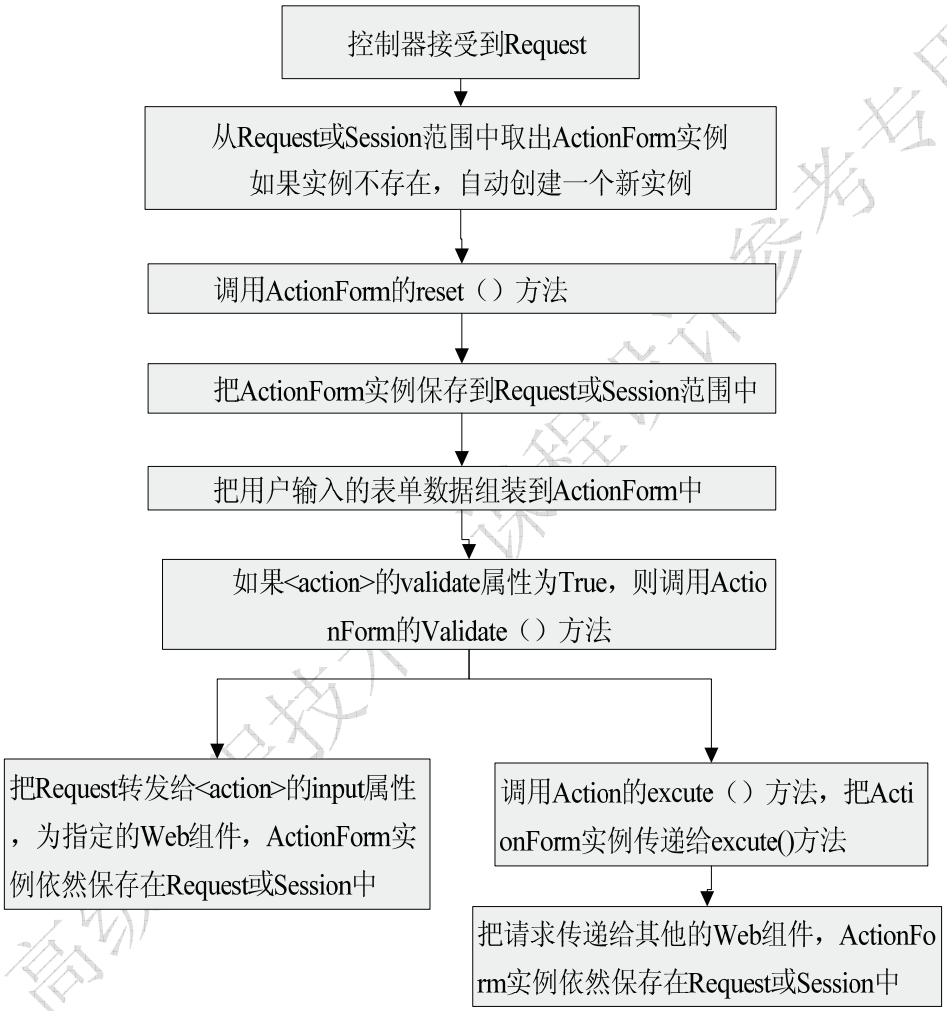


图 15.6　ActionForm 生存周期

下面以系统的登录视图实现为例，说明 ActionForm 的配置与实现过程（其他视图实现与此基本类似）：

- 创建相应 ActionForm

Struts 框架中定义的 ActionForm 类是抽象的，必须在应用中创建它的子类，来捕获具体的 HTML 表单数据，而 ActionForm Bean 中的属性 HTML 表单中的字段一一对应，具体代码如下：

```
public class LoginAction extends Action {
    public ActionForward execute(ActionMapping mapping,  ActionForm form,
```

```
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {
        try{
        user                        =                        (ProjectUser)
userdao.findByUsername(loginform.getUsername()).get(0);
        if(user.getPassword().equals(loginform.getPassword())){
            Cookie cookie = new Cookie("username", user.getUsername());
            if(user.getRole().equals("管理员")){
                cookie = new Cookie("login", "adminyes");
                response.addCookie(cookie);
                list = infodao.findAll();
                return mapping.findForward("manager");
            }
    else{
                cookie = new Cookie("login", "useryes");
                response.addCookie(cookie);
                list = infodao.findAll();
                return mapping.findForward("user");
            }
        }
    }
```

上述程序是用户登录时的处理。该 Action 根据用户的角色判断用户的映射方向。并会将一些重要的信息存入内置对象 Request 和 Cookie 中，一边后期使用。

- 配置 ActionForm

Struts 配置文件的<form-beans>元素用来配置所有的 ActionForm Bean。而<form-beans>元素可以包含多个<form-beans>子元素，而它即可代表单个的 ActionForm Bean.

```
<form-beans>
<form-bean name="LoginForm"  type=
          "com.yourcompany.struts.forms.LoginForm">
</form-bean>
…
</form-beans>
```

接下来为 ActionForm 配置 Action 映射。而 ActionForm 和 Action 允许是一对多的关系，同一个 ActionForm 可以和多个 Action 映射。在<action>元素中，name 和 scope 属性分别指定 ActionForm 的名字和范围，部分代码如下：

```
<action path="/Login"type="com.yourcompany.struts.actions.LoginAction
          "name="LoginForm">
    <forward name="user" path="/userworkbench.jsp"></forward>
    <forward name="manager" path="/manageworkbench.jsp"></forward>
    <forward name="failure" path="/login.jsp"></forward></action>
```

登录和功能菜单的 struts 视图，如图 15.7 所示。

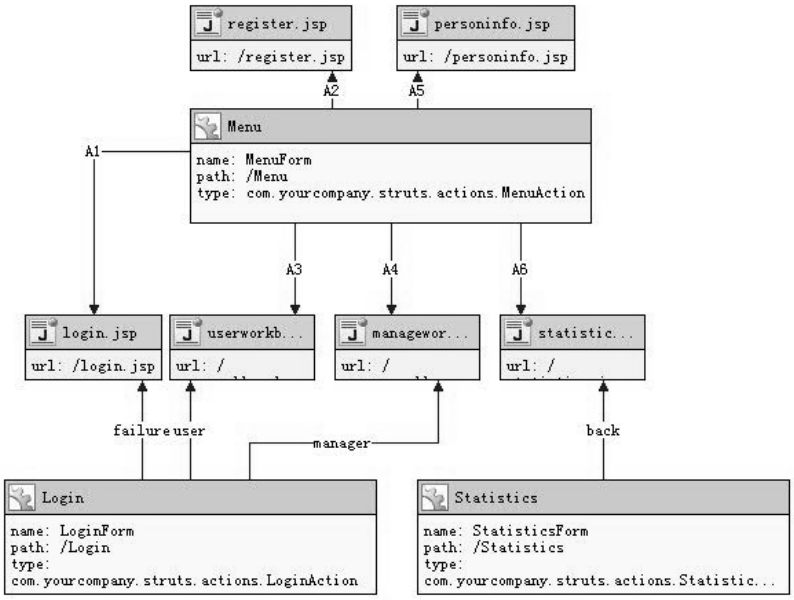


图 15.7　Struts 登录和功能菜单图

合同管理 Struts 视图，如图 15.8 所示。

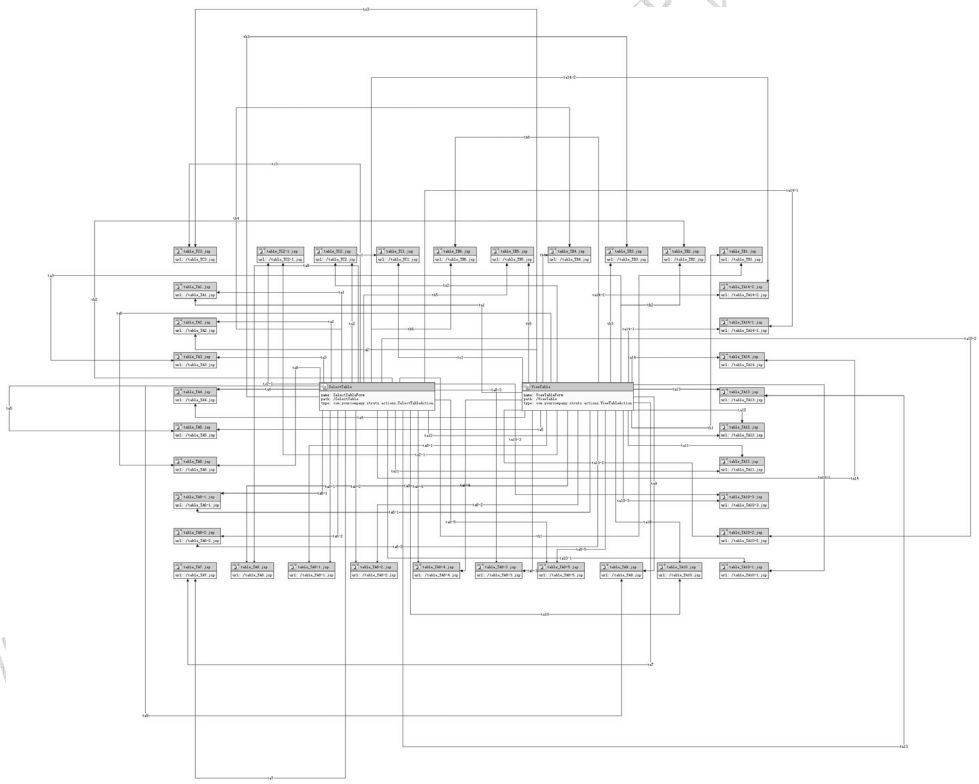


图 15.8　Struts 合同管理模块图

项目配置 Struts 视图，如图 15.9 所示。

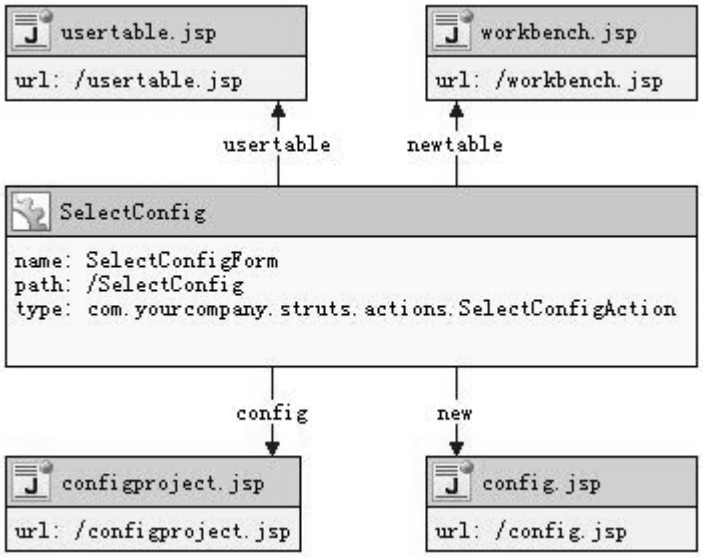


图 15.9　Struts 项目配置图.

全局配置 Struts 视图，如图 15.10：

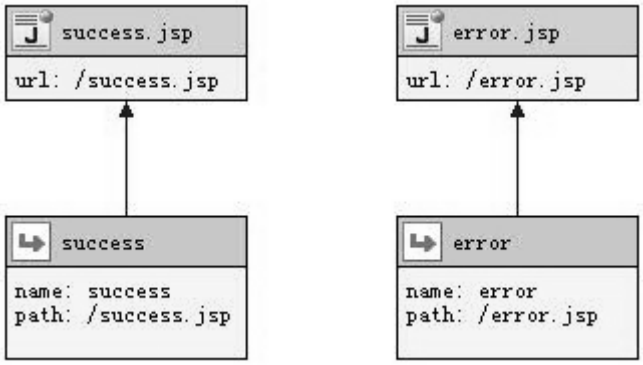


图 15.10　Struts 全局配置图

15.4.2　Struts 标签库

Struts 框架为表示层提供了丰富的 Taglib 标签库，本系统表示层应用 Taglib 标签代替传统 html 标签，充分利用 Struts 架构优势使业务逻辑与视图彻底分离。Struts 规定如果使用自定义标签，首先必须在 web.xml 文件中加入下列代码：

```
<taglib>
     <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
     <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
   </taglib>
   <taglib>
     <taglib-uri>/WEB-INF/struts-tiles.tld</taglib-uri>
     <taglib-location>/WEB-INF/struts-tiles.tld</taglib-location>
</taglib>
```

而在 JSP 中声明 Taglib 关键代码如下：

```
<%@taglib uri="/WEB-INF/struts-bean.tld"prefix="bean"%>
<%@taglib uri="/WEB-INF/struts-html.tld"prefix="html"%>
```

在以上代码中用到了 Struts 自带的 Taglib 标签库，其中 Struts HTML 标签用来在 JSP 文

件中产成 HTML 元素、协调表单处理，并将 JSP 页面（即视图页面）连接到 Struts 框架的其他部分，在 Struts 应用中提倡使用 Struts HTML 标签，这是因为这些标签可以和框架的其他组件更紧密的联系在一起。

15.4.3　Sitemesh 框架搭建

Sitemesh 是 J2EE 应用框架之一，用于提高页面的可维护性和复用性。它是通过创建一个包装对象，也就是装饰来包裹真实的对象。Sitemesh 的工作原理：一个请求到服务器后，如果该请求需要被 Sitemesh 装饰，服务器会先解释被请求的资源，然后根据配置文件获得用于该请求的装饰器，最后用装饰器装饰被请求资源，将结果一同返回给客户端浏览器。

Sitemesh 利用 Servlet 规范实现了一种页面过滤器。在页面被处理之后，返回 Web 浏览器之前，对页面做了一些附加的操作。

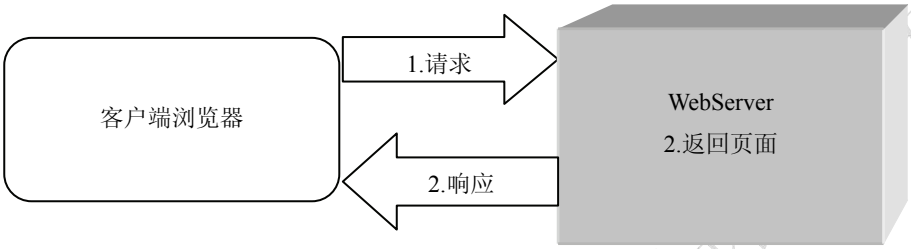


图 15.11　普通页面处理图.

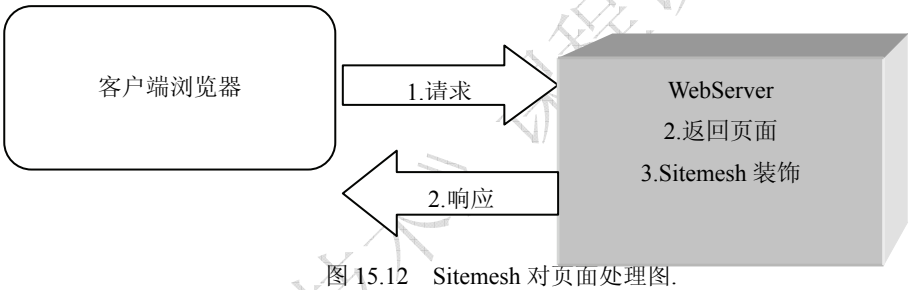


图 15.12　Sitemesh 对页面处理图.

- Sitemesh 的模板 frame.jsp

```
<html>
<head>
<title><decorator: title default="Welcome!" /></title>
<decorator: head />
</head>
<body onunload="RunOnUnload()">
 <script language="javascript">
 function RunOnBeforeUnload() {
  window.event.returnValue = '关闭浏览器将退出系统.';
  }
 function RunOnUnload() {
  Cookie cookie = new Cookie("login", "no");
  response.addCookie(cookie);
}
 </script>
<table style="width: 910px; height: 500px; " align="center">
    <tr>
        <td colspan="2"><img src="images/title left.jpg" height="140"></td>
    </tr>
    <tr>
```

```
        <td    ><div    style="position    :    relative    ;    "><%@    include
file="../left.jsp" %></div></td>
        <td height="370"><decorator: body /></td>
        </tr>
    </table>
    </body>
    </html>
```

其框架的显示，如图 15.13。

图 15.13　Frame 框架图

- Sitemesh 在配置

```
    <filter>
       <filter-name>sitemesh</filter-name>

<filter-class>com.opensymphony.module.sitemesh.filter.PageFilter</filter-cla
ss>
      </filter>
      <filter-mapping>
        <filter-name>sitemesh</filter-name>
        <url-pattern>/*</url-pattern>
      </filter-mapping>
```

通过对所选页面的过滤使得多有的页面都装饰成 frame 的形式，使得整体风格得以显现。

15.5　业务逻辑层实现

15.5.1　业务逻辑层的框架实现

- 登录业务逻辑实现

登录的业务逻辑分为两个方面，一个是信息的验证，另一个是匹配用户的角色进行不同的映射。在此期间，我使用了 Cookie 技术把用户登录与否的信息保存，用以判断是否登录。并应用 Request 传递用户的其他基础信息。在角色判断方面，通过读取用户的角色信息判断用户是否为管理员，并做出不同的映射。

部分代码如下：

```
if(user.getRole().equals("管理员")){
    cookie = new Cookie("login", "adminyes");
    response.addCookie(cookie);
    list = infodao.findAll();
    request.setAttribute("list", list);
    return mapping.findForward("manager"); }else{
     cookie = new Cookie("login", "useryes");
     response.addCookie(cookie);
     list = infodao.findAll();
    request.setAttribute("list", list);
     return mapping.findForward("user");
}
```

- 工程管理业务逻辑

工程管理方面，主要是工程的新建和工程的人员配置。在新建这块，通过调用 Hibernate 框架下的 DAO 来实现工程信息的存储。

相关代码如下：

```
ProjectForm pf = (ProjectForm)form;
        ProjectInfoDAO pfdao =  new ProjectInfoDAO();
        ProjectInfo info = new ProjectInfo(pf.getProjectname(),
                            ……pf.getEnd());
        pfdao.save(info);
        return mapping.findForward("success");
```

在工程人员配置方面，首先读取用户信息，把用户按角色分为四类（相关的建设单位），然后根据角色所属的单位分类显示。在配置好人员后，对配置信息进行存储。

- 用户权限管理业务逻辑

为用户的权限管理，系统做了两层过滤，一，通过读取用户的角色信息，与 role 表信息进行匹配，匹配成功的，读取其能够形式的合同。二、在能够读取的合同页面内，读取用户的所属公司信息，根据公司类别的不同，提供相应的可编辑区域。这样的权限处理业务，使得用户只能处理与其职位相关的合同，不能够越权操作。保护了合同的有效性。

相关代码如下：

- 一级权限过滤：

```
<%  for(int i=0; i<list.size(); i++){
     ProjectInfo form =(ProjectInfo)list.get(i);
if(request.getAttribute("companyname").equals(form.getProjectBuildcompany
())||request.getAttribute("companyname").equals(form.getProjectConstructcomp
any())
||request.getAttribute("companyname").equals(form.getProjectDesigncompany())
||request.getAttribute("companyname").equals(form.getProjectMonitorcompany()
) )
 %>
```

- 二级权限过滤（以 TA1 为例）：

```
<script type="text/javascript">
   function edit(){
```

```
var company = "<%=session.getAttribute("company").toString()%>";
if(company=="建设单位"){
document.getElementById('table_block4_1').disabled="";
…… }
if(company=="监理单位"){
document.getElementById('table_block2_1').disabled="";
……}
if(company=="施工单位"){
document.getElementById('table_head1').disabled="";
……}}
</script>
```

15.5.2　业务逻辑层的实现结果

根据上一小节对具体实现的实例介绍，系统软件开发者可以很方便的实现系统需求分析中所列举的功能。接下来以系统中合同添加实例实现为例，说明其实现结果。

首先对于用户来说，必须通过登录界面 login.jsp 进入系统主界面 login.jsp（如图 15.14），必须特别指出的是此用户必须是已经存在的用户，同时还具备添加合同的权限。

图 15.14　登录界面图

通过登录界面，普通用户进入界面如图 15.15 所示。

图 15.15　用户个人平台图.

通过选取相应的工程进入表格信息平台如图 15.16 所示。

图 15.16　表格信息平台图

通过新建合同，进入新建界面如图 15.17 所示。

图 15.17　新建界面图.

在此界面可以新建与工程有关的 36 个表格，下面以表 TA-1 为例如图 15.18 所示。

图 15.18　Ta-1 编辑界面图.

若登录时以管理员身份登录，则显示工程管理界面如图 15.19 所示。

图 15.19　工程管理界面图.

可以在继续进入工程新建界面（如图 15.20）和工程人员配置界面（如图 15.21）。

图 15.20　工程新建界面图

图 15.21　工程人员配置界面图.

15.6　数据持久层实现

15.6.1　DAO 模式介绍

很多的 J2EE 应用程序需要使用持久性数据（数据库管理系统、文件等）。不同的程序，持久性存储是各不相同的，并且用来访问这些不同的持久性存储机制的 API 也有很大的不同。如果应用程序要在不同的持久性存储间迁移，这些访问特定持久存储层的代码将面临重写。而数据访问对象（Data Access Object）模式为解决这一问题提供了一种相对高效的办法。系统使用 DAO 模式来抽象和封装所有对数据源的访问。DAO 管理着与数据源的连接以便检索和存储数据，它实现了用来操作数据源的访问机制。数据源可以是 DBMS、LDAP、File 等，依赖于 DAO 的业务组件为其客户端使用 DAO 提供更简单的接口。DAO 完全向客户端隐藏了数据源实现细节。由于当低层数据源实现变化时，DAO 向客户端提供的接口不会变化，所有该模式允许 DAO 调整到不同的存储模式，而不会影响其客户端或者业务组件。

- Business Object（业务对象）

代表数据客户端。正是该对象需要访问数据源以获取和存储数据。

- Data Access Object（数据访问对象）

作为该模式的主要对象，DAO 抽取该 BO 对象的低层数据访问实现，以保证对数据源的透明访问。BO 对象也可以把数据加载和存储操作委托给 DAO。

- Data Source（数据源）

代表数据源实现。数据源可以是 RDBMS 数据库、OODBMS、XML 文件等等。

- Value Object（值对象）

代表用做数据携带着的值对象。DAO 可以使用值对象来把数据返回给客户端。因为每个 BO 对应一个特殊的 DAO，因此需要建立 DAO 与底层关系的映射，例如 DAO 与数据库

表的映射，即前面所介绍的 ORM 映射。而由于很多第三方工具都具备自动生成 DAO 代码的功能，因此软件开发者只需选择合适的工具，根据工具自动生成相应的 DAO 代码即可，在本系统中采用的是最常用的 ORM 工具 Hibernate 自动生成其 DAO 代码，这样使程序员不必过多的关注 DAO 代码的原理，只需关注 Hibernate 自身的 ORM 实现即可。

15.6.2　持久层的 Hibernate 实现

在 Java Web 开发过程中，目前常用的几个持久层框架是 EJB 中的实体 Bean、JDO、Hibernate 和 Ibaties 等。Hibernate 是一种新的 ORM 映射工具，它不仅提供了从 Java 类到数据表的映射，也提供了数据查询和恢复等机制。相对于使用 JDBC 和 SQL 来手工操作数据库，使用 Hibernate，可以大大减少操作数据库的工作量。Hibernate 可以和多种 Web 服务器或者应用服务器良好集成，如今已经支持几乎所有的流行的数据库服务器。首先关注与本系统持久化相关包的介绍：

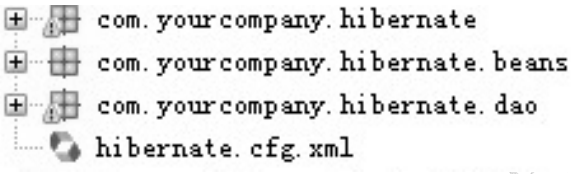


图 15.22　Hibernate 有关包的结构图

(1) com.yourcompany.hibernate.dao：DAO 模式的接口包；

(2) com.yourcompany.hibernate：持久化类的抽象文件；

(3) com.yourcompany.hibernate.beans：持久化类以及映射文件。

由于 DAO 代码为 Hibernate 自动生成，限于篇幅，在此不再详述。下面仅以新用户的添加为例说明如何使用 Hibernate 框架实现数据持久化的。

- 建立 Hibernate 的配置文件

为了使 Hibernate 框架真正的高效运行起来，必需为系统创建一个配置文件。Hibernate 同时支持 xml 格式的配置文件，以及传统的 properties 文件配置方式，不过在本系统中采用的是 xml 类型的配置文件。这是因为 xml 配置文件提供了更易读的结构和更强的配置能力，可以直接对映射文件加以配置，而在 properties 文件中则无法配置，必须通过代码中的硬编码方式加载相应的映射文件。在该配置文件中主要是进行 sessionFactory 配置（用于建立与数据库之间连接的配置），配置文件名默认为”hibernate.cfg.xml”或者（hibemate.properties），Hibemate 初始化期间会自动寻找这个文件，并读取其中的配置信息，为后期数据库操作做好准备。

在本系统中采用以 xml 格式文件创建一个 Hibernate 配置文件 hibernate.cfg.xml，则其代码如下所示：

```
    <hibernate-configuration>
        <session-factory>
            <property
name="dialect">org.hibernate.dialect.MySQLDialect</property>
            <property name="connection.url">
                jdbc: mysql: //localhost: 3306/projectdb?useUnicode=true&amp;
characterEncoding=utf-8&amp; mysqlEncoding=utf-8&amp; </property>
            <property name="connection.username">root</property>
            <property
name="connection.driver_class">org.gjt.mm.mysql.Driver</property>
            <property       name="myeclipse.connection.profile">JDBC       for
MySQL</property>
            <property name="userunicode">trueamp</property>
```

```
            <property name="characterEncoding">utf-8</property>
            <mapping
resource="com/yourcompany/hibernate/beans/ProjectTableTa1.hbm.xml" />
    ……
    </session-factory>
    </hibernate-configuration>
```

以上的 xml 配置文件代码表示了本系统与数据库连接的基本配置信息。正如代码注释中所说<property>的 name 为 dialect 的语句表示本系统数据库指定的方言为 MySQL 数据库管理系统。在这里要特别说明 Hibernate 几乎支持目前市场上所有常用的数据库系统，而且为其所支持的所有数据库系统配置了相应的 dialect，同时根据对应的 dialect 为系统提供了相应的数据库驱动，对应于上面的代码 name 值为"connection.driver_class"所表示的值就规定了本管理系统的数据库驱动值，一定要注意系统 dialect 与 driver 是对应的。代码中还规定了本系统的连接 URL，该属性的值表示了数据库服务器的服务端口号以及访问地址，同时还规定了系统的数据库名称为 projectdb，而系统程序在运行过程中，通过接下来的两个属性，可以得知本系统数据库的用户名为 root，密码为空，这样加上数据库名就可以使系统开发前台与数据库后台建立联系。

- 构建相应的数据库表的映射文件

对应于添加用户功能的数据库表为 project_user 表，其字段具体信息见第四章的数据库设计中。软件开发者首先创建与用户表 project_user 对应的映射 ProjectUser.hbm.xml，其具体代码如下：

```
<hibernate-mapping>
<class name="com.yourcompany.hibernate.beans.ProjectUser"
table="project_user" catalog="projectdb">
       <id name="id" type="java.lang.Integer">
          <column name="Id" />
          <generator class="identity" />
       </id>
       <property name="username" type="java.lang.String">
          <column name="username" />
       </class>
</hibernate-mapping>
```

映射文件是遵循 xml 文件的 DTD 定义格式，首先规定了文件的版本和字符编码。然后通过对文件中元素的值的设定与数据库表实现映射关系的。

(1) <hibernate-mapping>：用于规定 Hibernate 的映射关系，包含了几个常用属性，其中 schema 属性指明了这个映射的表所在的 schema 名称。default-cascade 属性指定了默认的级联风格可取值有 none、save、update。在 ProjectUser.hbm.xml 中，主要使用了 package 属性指定一个包前缀。

(2) <class>元素：用于定义一个持久化类，其中 table 属性值表明了此类与 project_user 数据表的对应关系。

(3) <id>：id 元素定义了属性到数据库表主键字段的映射，为该持久化类的实例生成唯一的标识，表明了表的主键 id 与持久化类的对象 ProjectUser 对应，而其中的<generator>属性用于指名主键的自增长类型。

(4) <property>：为持久化类声明了一个持久化的，JavaBean 风格的属性，同时指定了对象的类型和对应数据表的类型与大小等属性。

- 创建相应的持久类

在配置完 Hibernate 映射文件之后，可以直接通过工具生成对应于用户表的持久化类，同时也可以通过手动添加，在这里笔者采用手动添加。持久化类 ProjectUser.java 的部分代

码如下所示：

```
package com.yourcompany.hibernate.beans;
public abstract class AbstractProjectUser implements java.io.Serializable {
    /** default constructor */
    public AbstractProjectUser() {}
    public AbstractProjectUser(String username, String password,
            String realname, String company, String role, String email,
            String companyname) {
this.username = username;
        private Integer id;
    private String username;
    public Integer getId() {return this.id;}
    public void setId(Integer id) {this.id = id;}
}
```

以上四步就是使用 Hibernate 框架技术对系统进行持久化的一个典型过程，系统数据库中的其他表及其他部分的持久化操作均类似。

15.6.3　Struts 和 MySQL 中文乱码处理问题

Struts 与 Mysql 的中文处理问题是一件繁琐的事情。处理其中文问题必须对二者分别进行处理。

在 Struts 方面，需要添加一个表单过滤器 EncodingFilter.java，过滤提交的汉字符号，过滤代码为：

```
public class EncodingFilter extends javax.servlet.http.HttpServlet
implements Filter {
        FilterConfig filterConfig;
public void doFilter(ServletRequest request, ServletResponse response,
FilterChain filterChain) throws IOException, ServletException {
        MyRequest req = new MyRequest( (HttpServletRequest)request); //设置代
理请求
        filterChain.doFilter(req, response);   }
public void init(FilterConfig arg0) throws ServletException {
        this.filterConfig = arg0;  }}
class MyRequest extends javax.servlet.http.HttpServletRequestWrapper{//代
理请求类
       HttpServletRequest request;
       public MyRequest(HttpServletRequest request) {
        super(request);
        this.request = request; }
       public String[] getParameterValues(String name){
        String strs[] = super.getParameterValues(name);
        for(int i=0;i<strs.length;i++){strs[i] =this.myEncoding(strs[i]); }
        return strs; }
       private String myEncoding(String input){
        String output ="";
        try {
         output = new String(input.getBytes("ISO-8859-1"), "UTF-8");
//注意同页面的编码保持一致     }
catch (UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace(); }
```

```
        return output;  }
}
```

然后在 Web.xml 文件中对这个过滤器进行配置：

```
  <filter>
  <filter-name>encoding</filter-name>
  <filter-class>com.yourcompany.struts.EncodingFilter</filter-class>
 </filter>
 <filter-mapping>
  <filter-name>encoding</filter-name>
  <url-pattern>/*</url-pattern>
 </filter-mapping>
```

在 MySQL 方面，需在对数据库的连接字上加入相应的控制参数：

```
useUnicode=true&amp;characterEncoding=utf-8&amp;mysqlEncoding=utf-8&amp;
```

进过这两部处理后，能够实现中文的录入和打印。

小结

本章系统剖析大型建设项目资源合同管理系统的设计方案与具体实现的关键技术，具体体现软件工程核心思想中的各个实现环节：需求分析、存储模型及数据库设计、系统概要设计、基于 Struts 的三层架构（表示层、业务逻辑层、Hibernate 数据持久化层）。

思考与练习

1. 从完善的角度，对本章阐述的“大型建设项目的资源合同管理系统”，写出具体的改进意见，并实现它？
2. 从开发和维护的角度，谈谈开发基于 WEB 的系统的人员配置及分工，应如何合理进行，查阅最新相关理论，选择其中的一种开发模型，具体实现本章的系统，要求写好文档。