**上海电力学院**

**.NET 程序设计**



**题　　目:**  时间银行

**姓 名：** 黄海琨

**学 号：** 20131809

**院　　系：**　　 计算机科学与技术学院

**专业年级：**计算机科学与技术（电企方向）/2013级

1. **年 12 月 23日**

**目录**

[一、 设计题目 1](#_Toc438730952)

[1.1实验题目 1](#_Toc438730953)

[1.2问题描述 1](#_Toc438730954)

[1.3基本要求 1](#_Toc438730955)

[二、 需求分析 1](#_Toc438730956)

[2.1功能描述 1](#_Toc438730957)

[2.2 功能模块 2](#_Toc438730958)

[2.3数据流图 2](#_Toc438730959)

[三、 概要设计 3](#_Toc438730960)

[3.1 系统功能架构 3](#_Toc438730961)

[3.2 数据字典 4](#_Toc438730962)

[四、 系统设计 5](#_Toc438730963)

[4.1 系统总体设计 5](#_Toc438730964)

[4.2 系统数据模型设计 5](#_Toc438730965)

[4.3 系统实体联系图 5](#_Toc438730966)

[4.4 关系模型 6](#_Toc438730967)

[4.5 数据库表的设计 7](#_Toc438730968)

[五、 详细设计 8](#_Toc438730969)

[5.1 连接数据库语句 8](#_Toc438730970)

[5.2 用户登录 8](#_Toc438730971)

[5.3 用户注册 10](#_Toc438730972)

[5.4 用户信息修改 12](#_Toc438730973)

[5.5 用户主页 14](#_Toc438730974)

[5.6 显示用户的项目 16](#_Toc438730975)

[5.7 显示所有项目 17](#_Toc438730976)

[5.8 项目详情 19](#_Toc438730977)

[5.9 添加项目 21](#_Toc438730978)

[5.10 修改项目 23](#_Toc438730979)

[5.11 项目删除 26](#_Toc438730980)

[5.12 项目搜索 28](#_Toc438730981)

[5.13 项目内留言功能 30](#_Toc438730982)

[5.16 用户通知 31](#_Toc438730983)

[六、 测试 32](#_Toc438730984)

[6.1 测试 32](#_Toc438730985)

[6.2 系统存在不足之处 33](#_Toc438730986)

[七、总结 34](#_Toc438730987)

## 设计题目

### 1.1实验题目

时间银行

### 1.2问题描述

时间银行是用来保存你时间的一个网站，在这里人人都是平等的，在这里的交易都是通过时间来结算的，没有货币没有黄金，这里所有的都是平等的时间！

### 1.3基本要求

系统功能主要包括：

1. 用户的注册、登录、上传头像
2. 时间项目的发布、确认、查找
3. 项目留言和回复
4. 项目查找和推荐

## 需求分析

### 2.1功能描述

2.1.1用户的注册和登录

2.1.2用户留言和回复

2.1.3项目推荐和主页

2.1.4项目的发布、接收、确认、完成

### 2.2 功能模块

#### 2.2.1 用户模块

完成用户登录、注册、更改信息的功能。

#### 2.2.2 项目模块

项目创建、审核、搜索、参与、确认、核实、完成的功能，在每一个环节保证项目的可靠性。

#### 2.2.3 留言模块

留言模块在项目的各个环节都可以实施，来确保创建者与参与者的沟通，更好的完成项目。

### 2.3数据流图

系统分为普通用户和管理员两种，如图2-1所示

用户通过登录之后，提交项目申请给系统 ，系统提供可参与的项目列表，用户参加项目，完成后提交完成项目单，系统给项目创建者发出确认项目确定完成的通知，一旦项目创建者确认项目已完成，用户的账户时间会相应减少和增加。



图2-1 时间银行需求分析顶层图

如图2-2所示，用户在发布项目之后，系统将发布的项目信息存入可选项目表中，在接收项目中，系统从可选项目表取出项目放入已选项目表，在确认项目中，系统从可选项目表中取出项目放入已完成项目表中。在每个环节中，用户都可以进行留言功能。

图2-2 时间银行需求分析0层图

## 概要设计

### 3.1 系统功能架构

时间银行系统分为用户功能、项目功能、留言功能三大块，如图3-3所示

* 用户功能中有登录、注册、用户信息详情
* 项目功能中有添加项目、获取项目、确认项目、搜索项目
* 留言功能中有创建留言、查看留言、回复留言

图3-3 时间项目系统结构图

### 3.2 数据字典

数据字典是系统分析阶段的重要文档，它清楚地定义与详细地解释了数据流程图上不能详细表达的内容，同时它也是同用户交谈的工具。

下面详细列出本系统中的数据字典：

表3-4 网站用户数据字典

|  |  |
| --- | --- |
| 数据文件名： | 网站用户 |
| 简述： | 用户基本信息 |
| 数据文件组成： | 用户编号+用户账号+用户名+用户照片+用户年龄+用户性别+用户密码+用户分类+用户电话+用户邮箱+用户地址+用户兴趣+用户注册时间+用户权限+照片类型+用户时间 |

表3-5 项目信息数据字典

|  |  |
| --- | --- |
| 数据文件名： | 项目信息 |
| 简述： | 项目基本信息 |
| 数据文件组成： | 项目编号+项目名称+项目标签+项目创建者+项目时长+项目内容+项目执行者+项目创建时间+项目标记+项目结束时间 |

表3-6 留言信息数据字典

|  |  |
| --- | --- |
| 数据文件名： | 留言信息 |
| 简述： | 用户留言基本信息 |
| 数据文件组成： | 留言编号+发送留言者+送达留言者+留言创建时间+留言内容+留言标记 |

表3-7 管理员信息数据字典

|  |  |
| --- | --- |
| 数据文件名： | 管理员用户 |
| 简述： | 管理员用户基本信息 |
| 数据文件组成： | 管理员账户+管理员名称+管理员密码 |

## 系统设计

### 4.1 系统总体设计

ASP.Net 来创建这个网站，采用MVC架构来进行业务逻辑，Controller，View，Model和其他Bootstrap的空间搭建网页的后台和前端。系统将有用户模块、管理员模块、项目模块、留言模块组成。

### 4.2 系统数据模型设计

系统主要设计的实体有：用户、管理员、项目、留言

◾ 用户 属性有用户编号、用户账号、用户名、用户照片、用户年龄、用户性别、用户密码、用户分类、用户电话、用户邮箱、用户地址、用户兴趣、用户注册时间、用户权限、照片类型、用户时间

◾ 项目 属性有项目编号、项目名称、项目标签、项目创建者、项目时长、项目内容、项目执行者、项目创建时间、项目标记、项目结束时间

◾ 留言属性有留言编号、发送留言者、送达留言者、留言创建时间、留言内容、留言标记

◾ 管理员 属性有管理员账户、管理员名称、管理员密码

### 4.3 系统实体联系图

系统中的实体关系如图4-1所示

* 一个用户可以创建多个项目，一个项目只能有一个创建者和实施者，因此用户和项目的关系是一对多的关系。
* 一个用户可以创建多个留言，一个留言只能有一个发送者和接受者，因此用户和留言的关系是一对多的关系

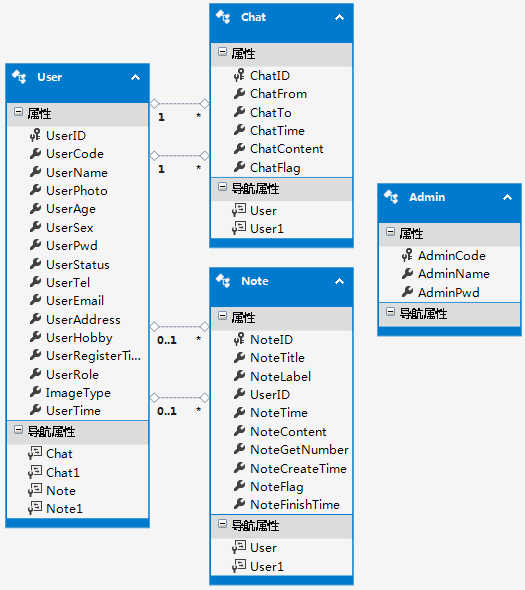


图4-1 实体联系ER图

### 4.4 关系模型

◾ 用户 （用户编号、用户账号、用户名、用户照片、用户年龄、用户性别、用户密码、用户分类、用户电话、用户邮箱、用户地址、用户兴趣、用户注册时间、用户权限、照片类型、用户时间）

◾ 项目 （项目编号、项目名称、项目标签、项目创建者、项目时长、项目内容、项目执行者、项目创建时间、项目标记、项目结束时间）

◾ 留言（留言编号、发送留言者、送达留言者、留言创建时间、留言内容、留言标记）

◾ 管理员（管理员账户、管理员名称、管理员密码）

### 4.5 数据库表的设计

系统的数据库表有用户表、项目表、留言表和管理员表

表4-2 用户表User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型 | 是否为空 | 是否为主键 |
| UserID | 用户编号 | int | FALSE | TRUE |
| UserCode | 用户账号 | nchar(10) | FALSE | FALSE |
| UserName | 用户名 | nchar(10) | FALSE | FALSE |
| UserPhoto | 用户照片 | image | TRUE | FALSE |
| UserAge | 用户年龄 | int | TRUE | FALSE |
| UserSex | 用户性别 | nchar(10) | TRUE | FALSE |
| UserPwd | 用户密码 | nvarchar(MAX) | FALSE | FALSE |
| UserStatus | 用户分类 | nchar(10) | TRUE | FALSE |
| UserTel | 用户电话 | nchar(10) | TRUE | FALSE |
| UserEmail | 用户邮箱 | nchar(10) | FALSE | FALSE |
| UserAddress | 用户地址 | nchar(10) | TRUE | FALSE |
| UserHobby | 用户兴趣 | nchar(10) | TRUE | FALSE |
| UserRegisterTime | 用户注册时间 | datetime | TRUE | FALSE |
| UserRole | 用户权限 | int | TRUE | FALSE |
| ImageType | 照片类型 | nvarchar(50) | TRUE | FALSE |
| UserTime | 用户时间 | int | TRUE | FALSE |

表4-3 项目表Note

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型 | 是否为空 | 是否为主键 |
| NoteID | 项目编号 | int | FALSE | TRUE |
| NoteTitle | 项目名称 | nvarchar(MAX) | FALSE | FALSE |
| NoteLabel | 项目标签 | nvarchar(MAX) | TRUE | FALSE |
| UserID | 项目创建者 | int | TRUE | FALSE |
| NoteTime | 项目时长 | int | TRUE | FALSE |
| NoteContent | 项目内容 | ntext | TRUE | FALSE |
| NoteGetNumber | 项目执行者 | int | TRUE | FALSE |
| NoteCreateTime | 项目创建时间 | datetime | TRUE | FALSE |
| NoteFlag | 项目标记 | int | TRUE | FALSE |
| NoteFinishTime | 项目结束时间 | datetime | TRUE | FALSE |

表4-4 留言表Chat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型 | 是否为空 | 是否为主键 |
| ChatID | 留言编号 | int | FALSE | TRUE |
| ChatFrom | 发送留言者 | int | FALSE | FALSE |
| ChatTo | 送达留言者 | int | FALSE | FALSE |
| ChatTime | 留言创建时间 | datetime | FALSE | FALSE |
| ChatContent | 留言内容 | ntext | FALSE | FALSE |
| ChatFlag | 留言标记 | int | TRUE | FALSE |

表4-5 管理员表Admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型 | 是否为空 | 是否为主键 |
| AdminCode | 管理员账户 | nchar(10) | FALSE | TRUE |
| AdminName | 管理员名称 | nchar(10) | FALSE | FALSE |
| AdminPwd | 管理员密码 | nchar(10) | FALSE | FALSE |

## 详细设计

### 5.1 连接数据库语句

<connectionStrings>

<add name="TimeProContext" connectionString="data source=(localdb)\v12.0;initial catalog=Timeweb;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />

</connectionStrings>

### 5.2 用户登录

用户登录的时候查询数据库用户表中的数据，看用户是否存在，如果用户数据存在，把用户的数据存入Session中，为之后的功能服务，若没有，则保持登录界面，如图5-1所示算法。



图5-1 登录过程流程图

**代码：**

[HttpGet]

public ActionResult Login()

{

return View();

}

[HttpPost]

public ActionResult Login([Bind(Include = "UserCode,UserPwd")] User user)

{

var result = from v in db.User

where v.UserCode == user.UserCode && v.UserPwd == user.UserPwd

select v;

if (result.FirstOrDefault() != null)

{

if (Session["username"] != "" && Session["password"] != "")

{

this.Session["username"] = user.UserCode;

this.Session["password"] = user.UserPwd;

}

Session.Timeout = 30;

return RedirectToAction("../Users/UsersIndex");

}

else

{

return Content("<script >alert('账号或密码有误！');history.go(-1)</script >", "text/html");

}

return RedirectToAction("../Users/Login");

}

**运行页面：**

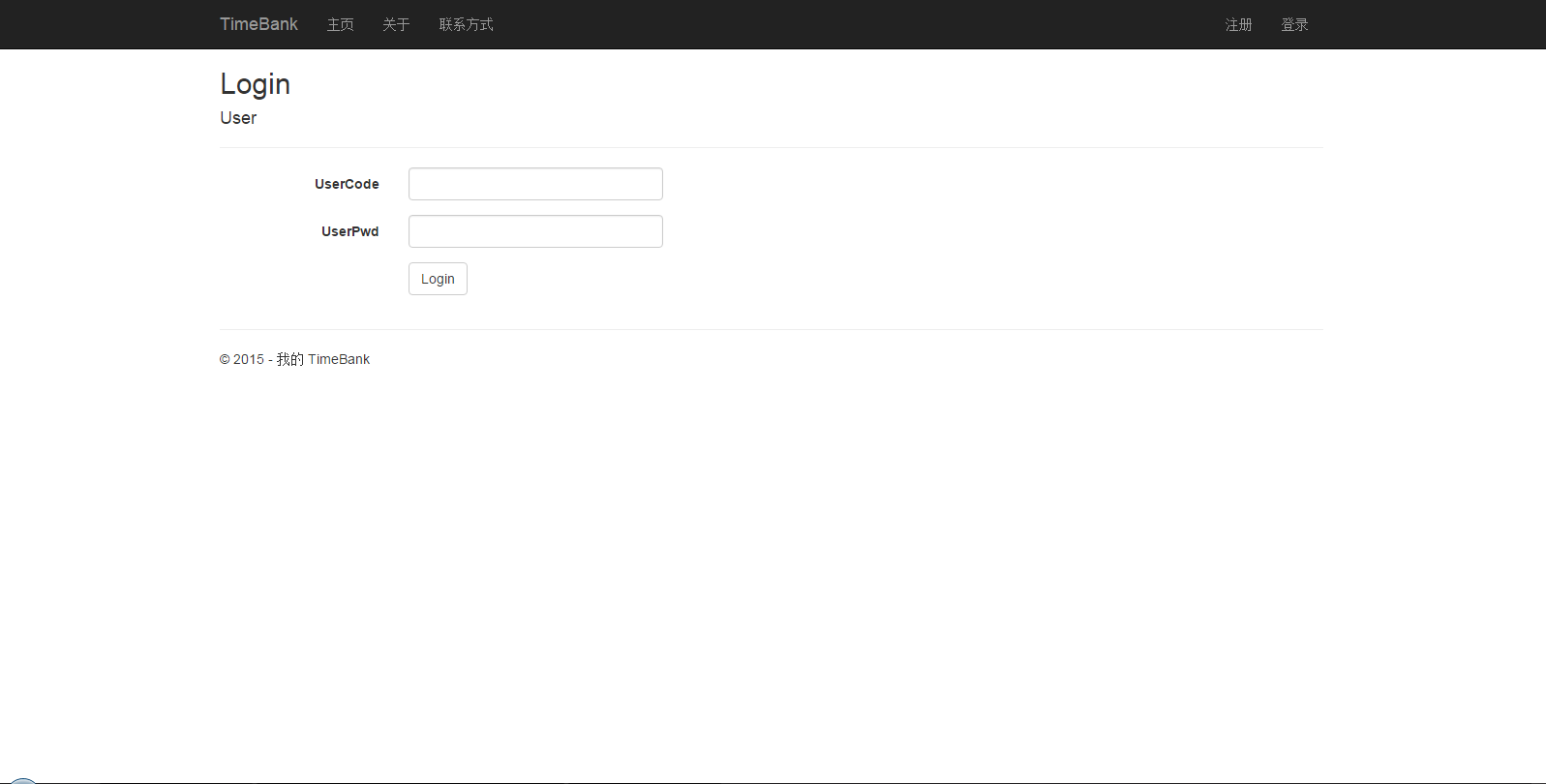


图5-2 登录页面

### 5.3 用户注册

用户注册输入用户名、昵称、密码和邮箱，系统判断用户名是否已在数据库用户表中存在，如果存在给出提示进行更改，如果不存在则成功注册跳转到登录界面，如图5-3所示算法。



图5-3 注册功能流程图

**代码：**

public ActionResult Register()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Register([Bind(Include = "UserID,UserCode,UserName,UserPhoto,UserAge,UserSex,UserPwd,UserStatus,UserTel,UserEmail,UserAddress,UserHobby,UserRegisterTime,UserRole,ImageType,UserTime")] User user)

{

var result = from v in db.User

where v.UserCode == user.UserCode

select v;

if (result.FirstOrDefault() == null)

{

if (ModelState.IsValid)

{

user.UserTime = 10;

user.UserRole = 0;

user.UserRegisterTime = System.DateTime.Now;

db.User.Add(user);

db.SaveChanges();

return RedirectToAction("../Users/Login");

}

}

else

{

return Content("<script >alert('账号已存在！');history.go(-1)</script >", "text/html");

}

return View(user);

}

**运行页面：**

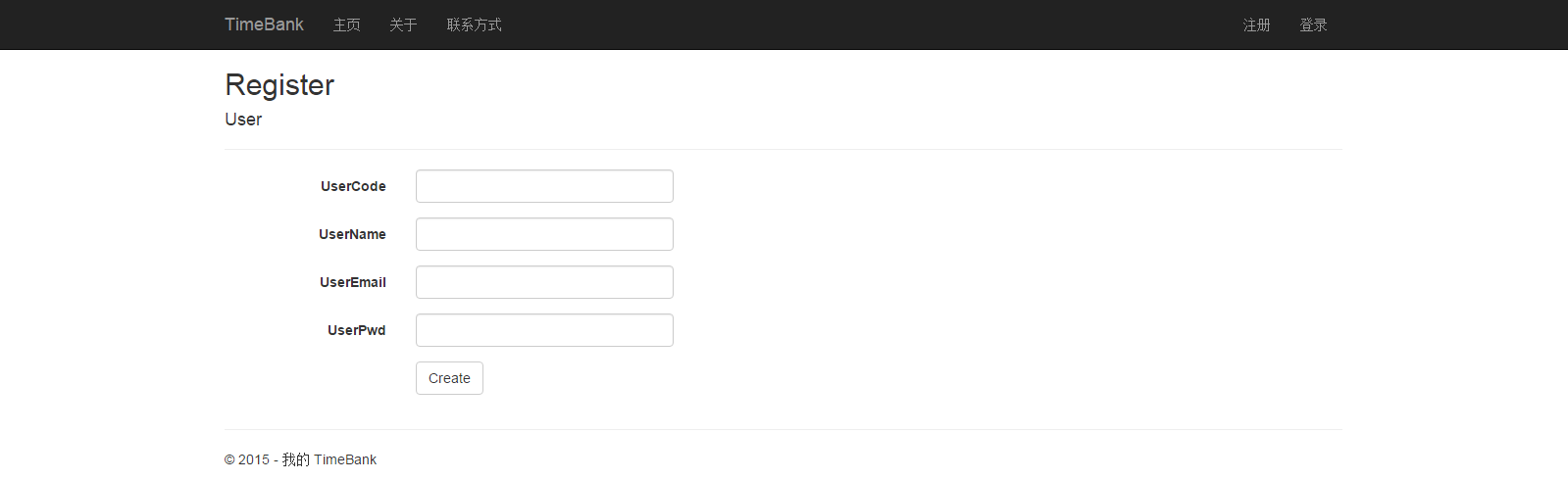


图5-4 注册页面

### 5.4 用户信息修改

在用户信息修改时，可以添加用户头像，选择用户头像，确定图片类型，保存到数据库。确定用户要修改的信息，这里采用ViewModel类型进行多数据的传输，算法如图5-5所示。



图5-5 用户信息修改功能流程图

**代码：**

[Filter.Filter.LoginFilter]

public ActionResult UsersEdit()

{

String usercode = Session["username"].ToString();

int id = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

id = result.UserID;

}

}

User user = db.User.Find(id);

if (user == null)

{

return HttpNotFound();

}

return View(user);

}

[HttpPost]

[ValidateAntiForgeryToken]

[Filter.Filter.LoginFilter]

public ActionResult UsersEdit(User user, HttpPostedFileBase image)

{

if (ModelState.IsValid)

{

User UserModel = db.User.Find(user.UserID);

UserModel.UserCode = user.UserCode;

UserModel.UserName = user.UserName;

UserModel.UserAge = user.UserAge;

UserModel.UserSex = user.UserSex;

UserModel.UserPwd = user.UserPwd;

UserModel.UserStatus = user.UserStatus;

UserModel.UserTel = user.UserTel;

UserModel.UserEmail = user.UserEmail;

UserModel.UserAddress = user.UserAddress;

UserModel.UserHobby = user.UserHobby;

if (image != null)

{

user.ImageType = image.ContentType;

user.UserPhoto = new byte[image.ContentLength]; image.InputStream.Read(user.UserPhoto, 0, image.ContentLength);

UserModel.UserPhoto = user.UserPhoto;

UserModel.ImageType = user.ImageType;

}

db.Entry(UserModel).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("UsersIndex");

}

return View(user);

}

**运行页面：**

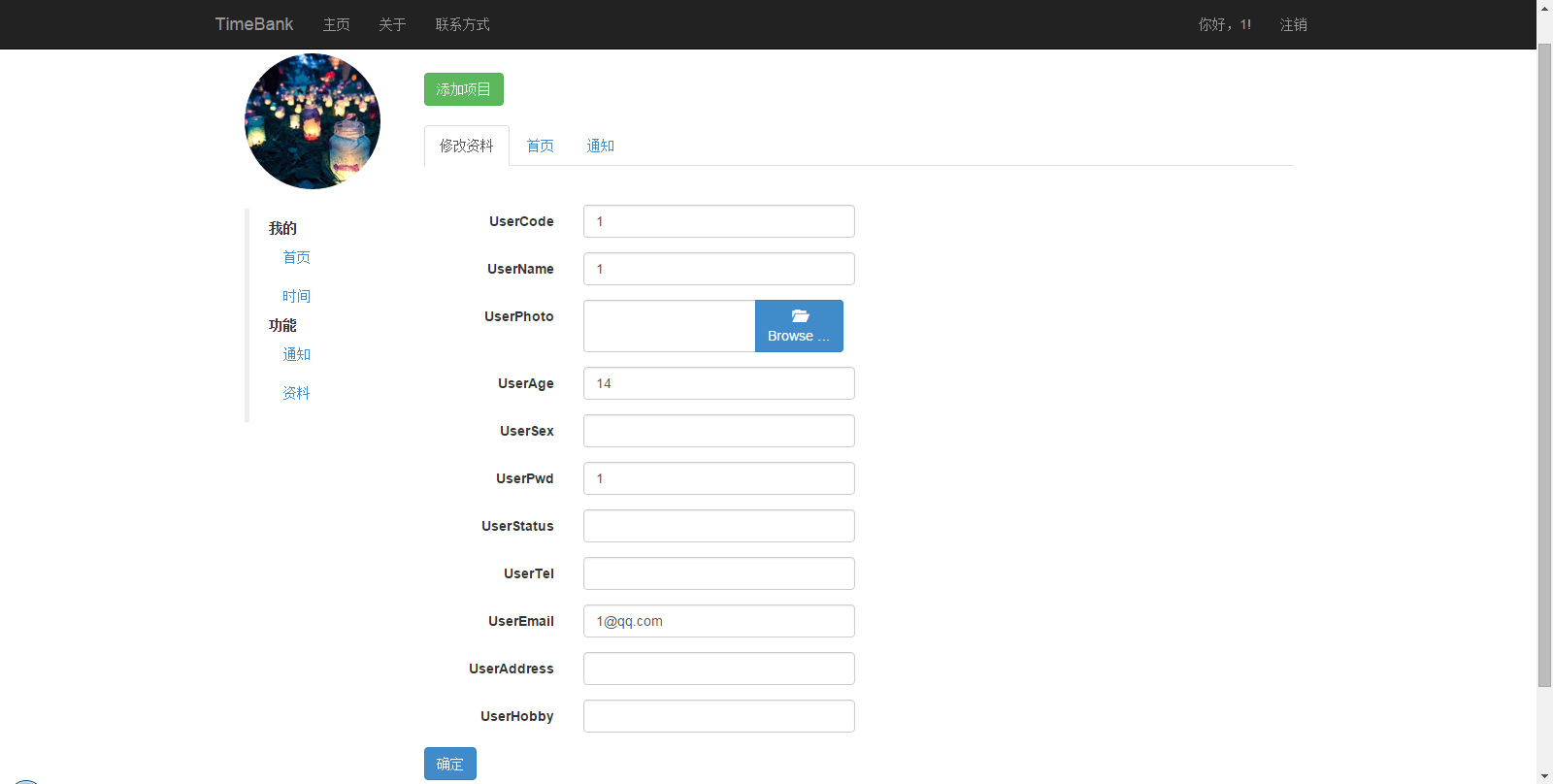


图5-6 用户信息修改页面

### 5.5 用户主页

提示用户时间、留言信息、推荐项目

推荐项目根据用户兴趣相同进行推荐，若未填写兴趣，系统默认按照最新项目进行推荐。



图5-7 用户主页功能流程图

**代码：**

[Filter.Filter.LoginFilter]

public ActionResult UsersIndex()

{

String usercode = Session["username"].ToString();

int id = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

id = result.UserID;

}

}

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

User user = db.User.Find(id);

if (user == null)

{

return HttpNotFound();

}

var note = (from b in db.Note.Include(n => n.User) where b.UserID != user.UserID && b.NoteFlag == 0 orderby b.NoteCreateTime descending select b).Take(3);

int chat = (from b in db.Chat where b.ChatTo == user.UserID && b.ChatFlag == 0 select b).Count();

ViewModel model = new ViewModel();

model.UserModel = user;

model.NoteModel = note.ToList();

model.temp = chat;

return View(model);

}

**运行页面：**



图5-8 用户主页页面

### 5.6 显示用户的项目

用户项目列出用户所建的项目，里面分为未有人参与、有人参与、等待确认、已完成项目的分类。



图5-9 项目标签判定功能流程图

**代码：**

[Filter.Filter.LoginFilter]

public ActionResult NotesIndex()

{

String usercode = Session["username"].ToString();

int nid = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

nid = result.UserID;

}

}

User user = db.User.Find(nid);

var note = from b in db.Note.Include(n => n.User) where b.UserID == user.UserID select b;

ViewModel model = new ViewModel();

model.UserModel = user;

model.NoteModel = note.ToList();

return View(model);

}

**运行页面：**

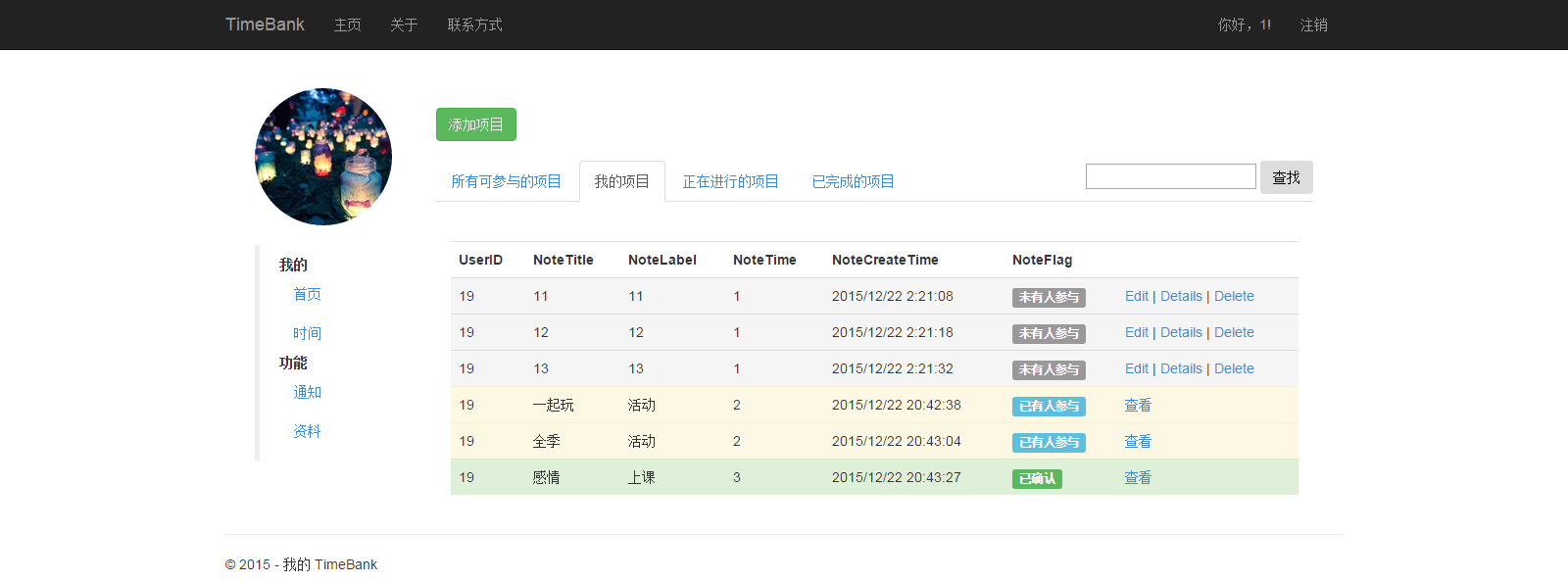


图5-10 项目页面

### 5.7 显示所有项目

此页面显示用户可以参与的所有项目列表

**代码：**

[Filter.Filter.LoginFilter]

public ActionResult NotesAll()

{

String usercode = Session["username"].ToString();

int nid = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

nid = result.UserID;

}

}

User user = db.User.Find(nid);

var note = from b in db.Note.Include(n => n.User) where b.UserID != user.UserID && b.NoteFlag == 0 orderby b.NoteCreateTime descending select b;

ViewModel model = new ViewModel();

model.UserModel = user;

model.NoteModel = note.ToList();

return View(model);

}

**运行页面：**

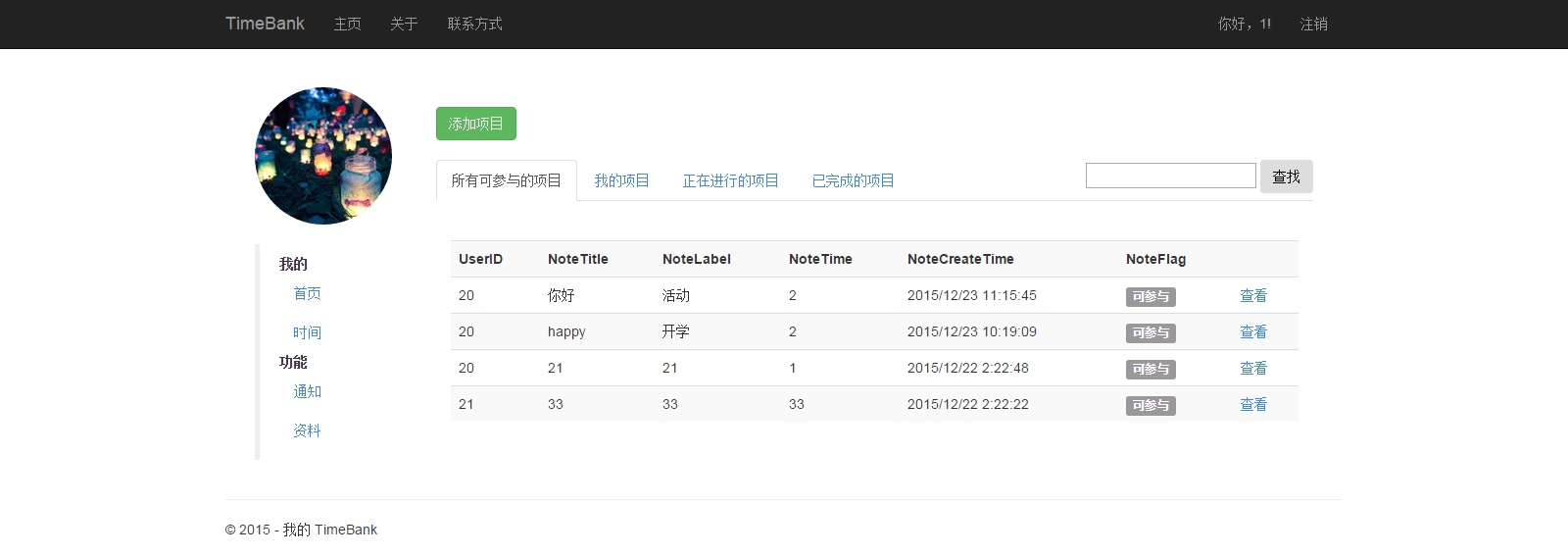


图5-11 所有项目页面

### 5.8 项目详情

通过读取项目ID找到项目详情信息

**代码：**

[Filter.Filter.LoginFilter]

public ActionResult NotesAllDetails(int? id)

{

String usercode = Session["username"].ToString();

int nid = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

nid = result.UserID;

}

}

User user = db.User.Find(nid);

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Note note = db.Note.Find(id);

if (note == null)

{

return HttpNotFound();

}

ViewModel1 model = new ViewModel1();

model.UserModel = user;

model.NoteModel = note;

return View(model);

}

[HttpPost]

[ValidateInput(false)]

[ValidateAntiForgeryToken]

[Filter.Filter.LoginFilter]

public ActionResult NotesAllDetails(ViewModel1 note){

String usercode = Session["username"].ToString();

int nid = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

nid = result.UserID;

}

}

User user = db.User.Find(nid);

Note NoteModel = db.Note.Find(note.NoteModel.NoteID);

if (note.NoteModel.UserID == user.UserID)

{

return Content("<script >alert('不能选择自己的项目！');history.go(-1)</script >", "text/html");

}

else

{

NoteModel.NoteGetNumber = user.UserID;

NoteModel.NoteFlag = 1;

if (ModelState.IsValid)

{

db.Entry(NoteModel).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("../Notes/NotesIng");

}

ViewBag.UserID = new SelectList(db.User, "UserID", "UserCode", note.UserModel.UserID);

}

return View(note);

}

**运行界面：**

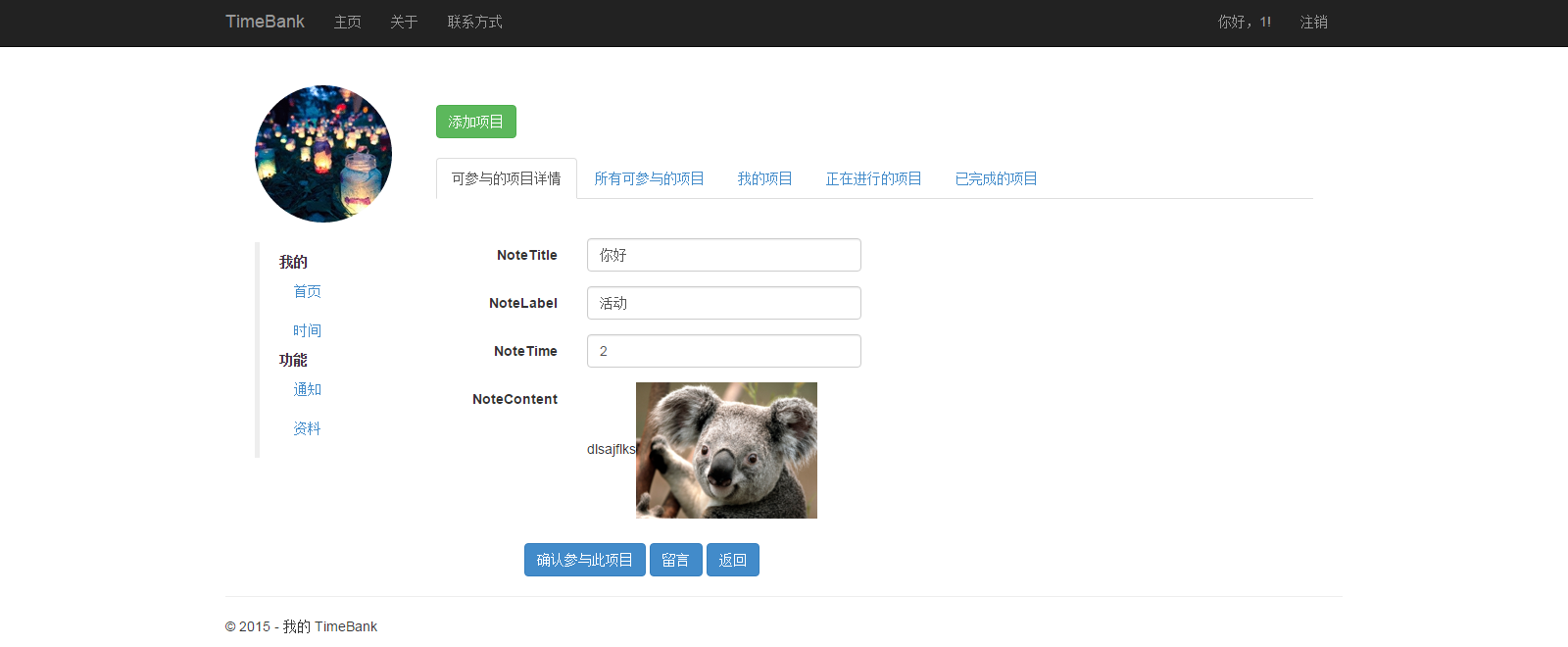


图5-12 项目详情页面

### 5.9 添加项目

添加项目，点击确定时候把图片信息用二进制方式存入数据库



图5-13 添加项目过程流程图

**代码：**

[Filter.Filter.LoginFilter]

public ActionResult NotesCreate()

{

String usercode = Session["username"].ToString();

int id = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

id = result.UserID;

}

}

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

User user = db.User.Find(id);

if (user == null)

{

return HttpNotFound();

}

Note note = new Note();

ViewModel1 model = new ViewModel1();

model.UserModel = user;

model.NoteModel = note;

return View(model);

}

[HttpPost]

[ValidateInput(false)]

[ValidateAntiForgeryToken]

[Filter.Filter.LoginFilter]

public ActionResult NotesCreate(ViewModel1 note)

{

String usercode = Session["username"].ToString();

int id = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

id = result.UserID;

}

}

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

User user = db.User.Find(id);

if (user == null)

{

return HttpNotFound();

}

note.NoteModel.NoteFlag = 0;

note.NoteModel.UserID = user.UserID;

note.NoteModel.NoteCreateTime = System.DateTime.Now;

if (ModelState.IsValid)

{

db.Note.Add(note.NoteModel);

db.SaveChanges();

return RedirectToAction("../Notes/NotesIndex");

}

ViewBag.UserID = new SelectList(db.User, "UserID", "UserCode", note.NoteModel.UserID);

ViewModel1 model = new ViewModel1();

model.UserModel = user;

model.NoteModel = note.NoteModel;

return View(model);

}

**运行页面：**

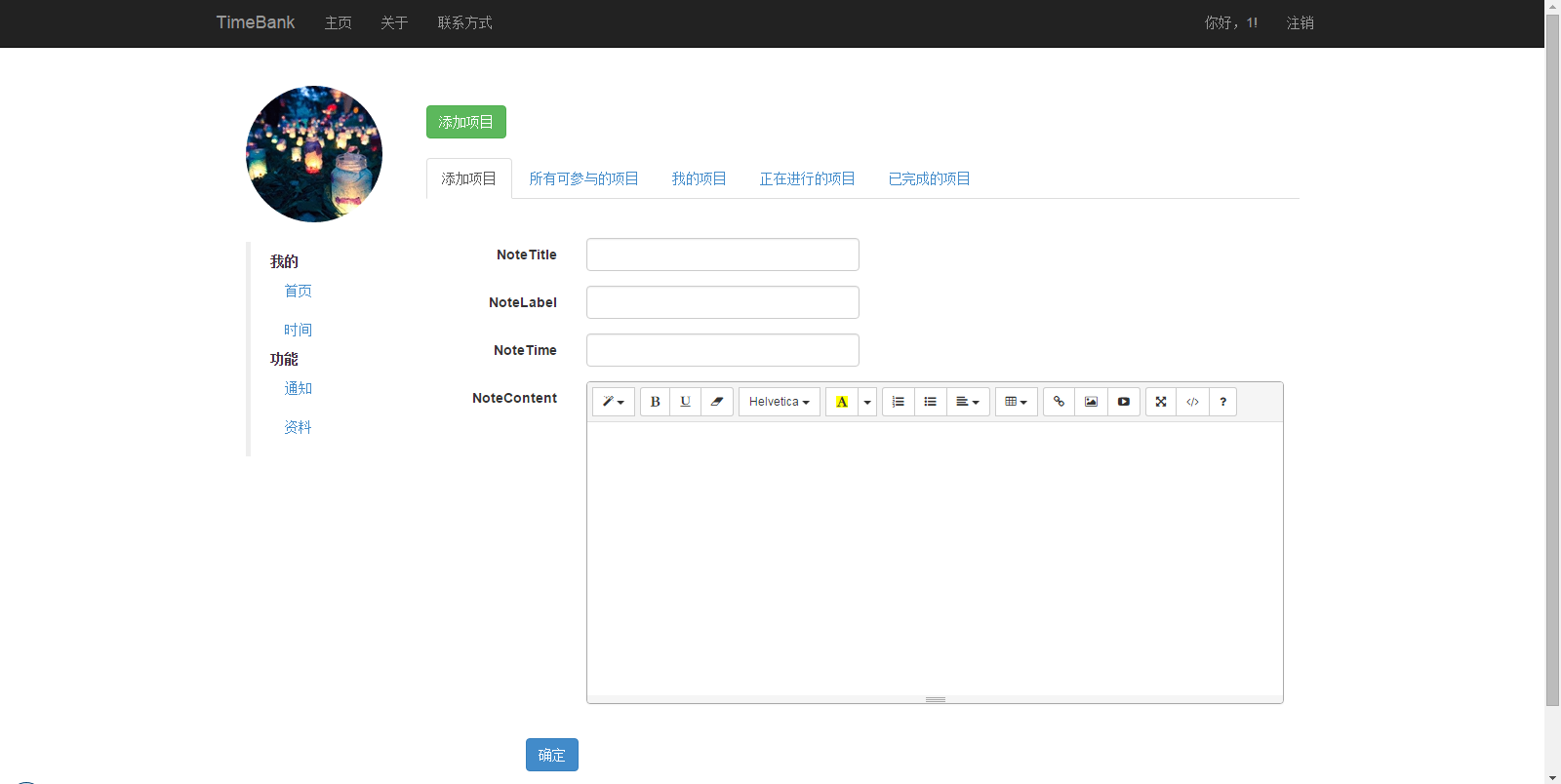


图5-14 添加项目页面

### 5.10 修改项目

根据项目ID找到项目，呈现项目内容，确定修改项目，表单提交



图5-15 修改项目内容流程图

**代码：**

[ValidateInput(false)]

[Filter.Filter.LoginFilter]

public ActionResult NotesEdit(int? id)

{

String usercode = Session["username"].ToString();

int nid = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

nid = result.UserID;

}

}

User user = db.User.Find(nid);

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Note note = db.Note.Find(id);

if (note == null)

{

return HttpNotFound();

}

ViewBag.UserID = new SelectList(db.User, "UserID", "UserCode", note.UserID);

ViewModel1 model = new ViewModel1();

model.UserModel = user;

model.NoteModel = note;

return View(model);

}

[HttpPost]

[ValidateInput(false)]

[ValidateAntiForgeryToken]

[Filter.Filter.LoginFilter]

public ActionResult NotesEdit(ViewModel1 note)

{

String usercode = Session["username"].ToString();

int nid = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

nid = result.UserID;

}

}

User user = db.User.Find(nid);

Note NoteModel = db.Note.Find(note.NoteModel.NoteID);

NoteModel.NoteTitle = note.NoteModel.NoteTitle;

NoteModel.NoteLabel = note.NoteModel.NoteLabel;

NoteModel.NoteTime = note.NoteModel.NoteTime;

NoteModel.NoteContent = note.NoteModel.NoteContent;

if (ModelState.IsValid)

{

db.Entry(NoteModel).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("../Notes/NotesIndex");

}

ViewBag.UserID = new SelectList(db.User, "UserID", "UserCode", note.UserModel.UserID);

return View(note.NoteModel);

}

**运行页面：**

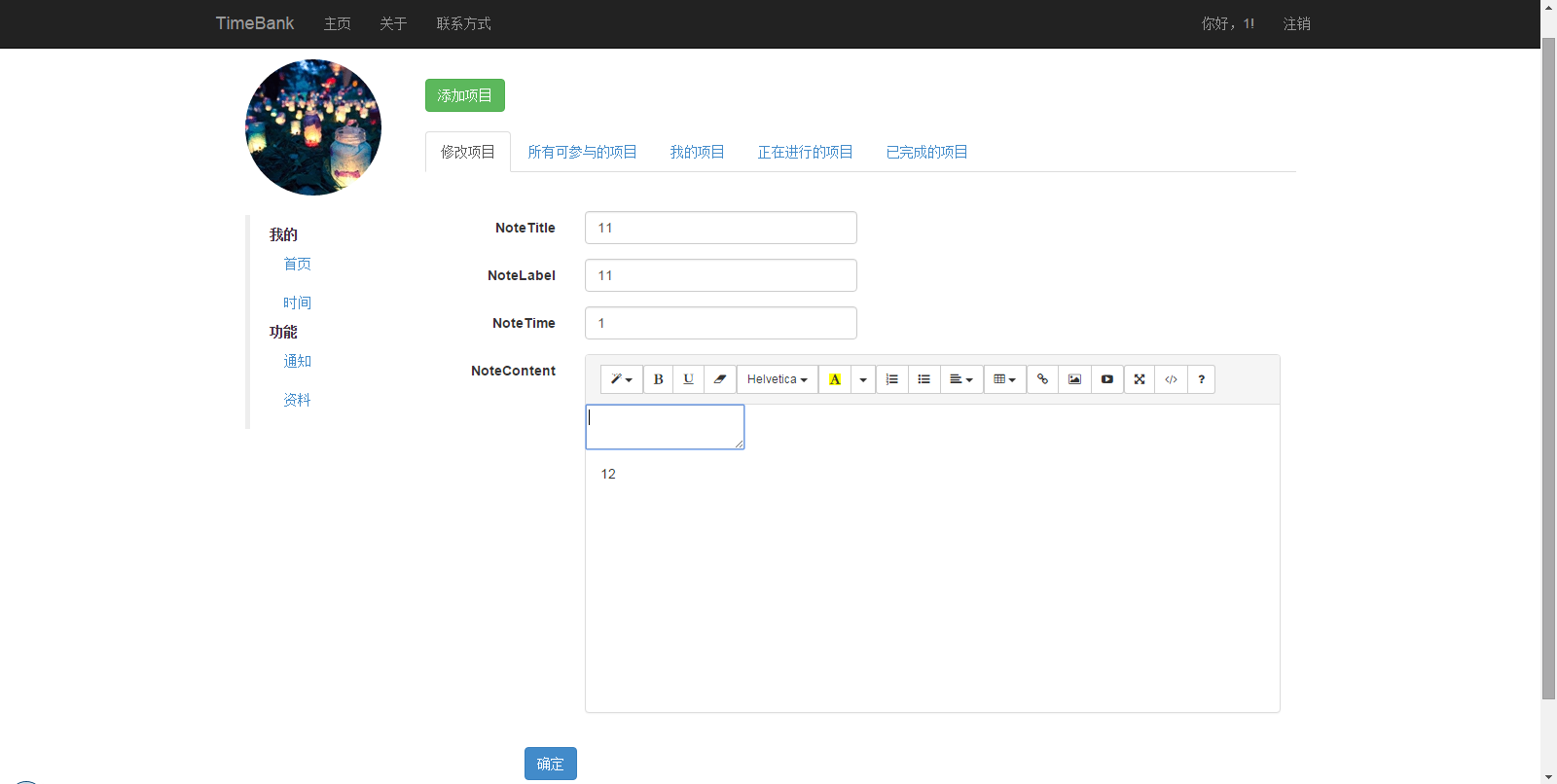


图5-16 修改项目页面

### 5.11 项目删除

确定项目ID，询问是否删除，在数据库中删除



图5-17 项目删除流程图

**代码：**

[Filter.Filter.LoginFilter]

public ActionResult NotesDelete(int? id)

{

String usercode = Session["username"].ToString();

int nid = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

nid = result.UserID;

}

}

User user = db.User.Find(nid);

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Note note = db.Note.Find(id);

if (note == null)

{

return HttpNotFound();

}

ViewBag.UserID = new SelectList(db.User, "UserID", "UserCode", note.UserID);

ViewModel1 model = new ViewModel1();

model.UserModel = user;

model.NoteModel = note;

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

[Filter.Filter.LoginFilter]

public ActionResult DeleteConfirmed(ViewModel1 model)

{

int id = model.NoteModel.NoteID;

Note note = db.Note.Find(id);

db.Note.Remove(note);

db.SaveChanges();

return RedirectToAction("../Notes/NotesIndex");

}

**运行页面：**

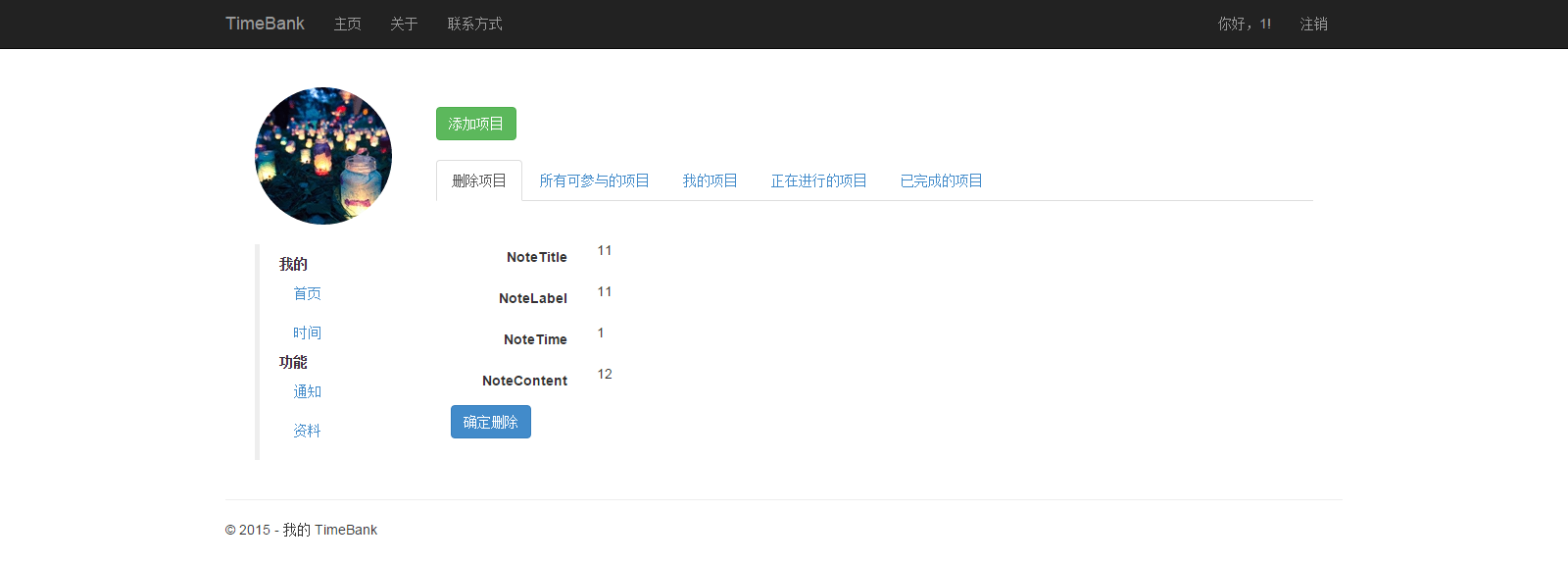


图5-18 项目删除页面

### 5.12 项目搜索

在所有列表的查询窗口中输入内容，系统会值项目的标题、标签、内容中模糊查询并返回。



图5-19 项目查询

**代码：**

[Filter.Filter.LoginFilter]

public ActionResult NotesFind(string search)

{

var query = db.Note.Where(m => m.NoteTitle.Contains(search) || m.NoteLabel.Contains(search) || m.NoteContent.Contains(search));

String usercode = Session["username"].ToString();

int id = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User where a.UserCode == usercode select a;

foreach (var result in Result)

{

id = result.UserID;

}

}

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

User user = db.User.Find(id);

if (user == null)

{

return HttpNotFound();

}

ViewModel model = new ViewModel();

model.NoteModel = query.ToList();

model.UserModel = user;

return View(model);

}

**运行页面：**

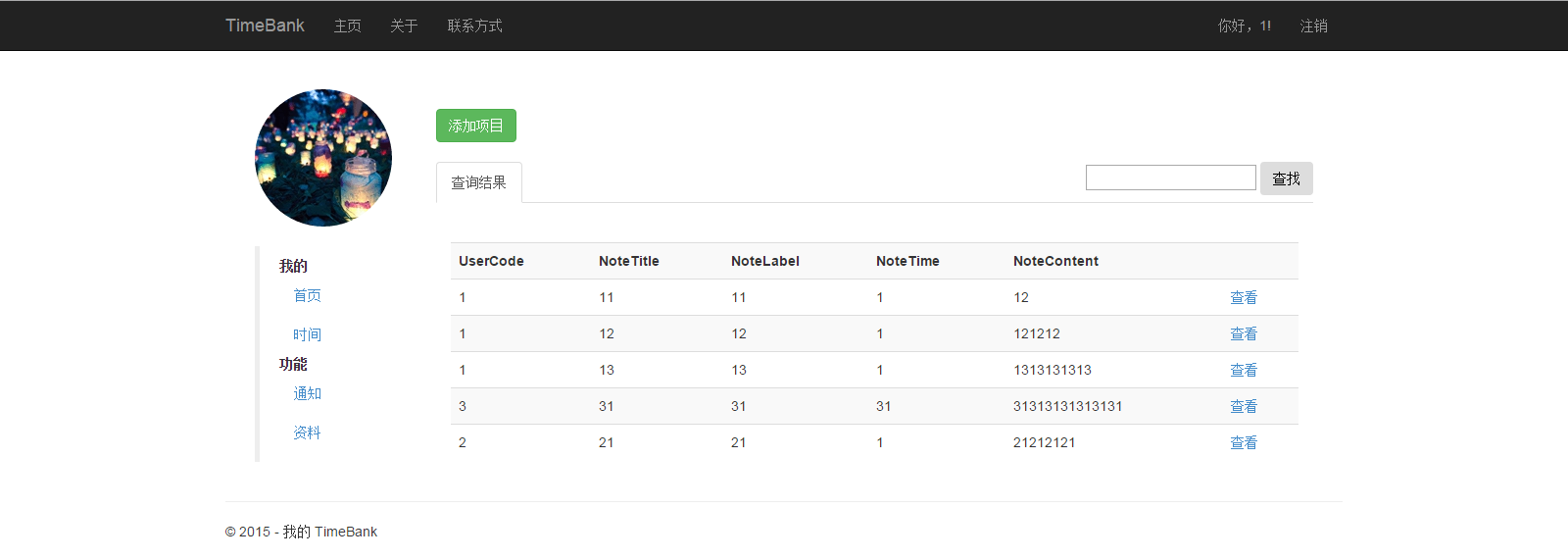


图5-20 项目搜索页面

### 5.13 项目内留言功能

在项目详情页面上，通过点击回复功能，在页面上跳出留言对话框，系统自动确定发送者和接收者。



图5-21 留言功能流程图

**代码：**

public ActionResult ChatsCreate(ViewModel1 model)

{ model.ChatModel.ChatFrom = model.UserModel.UserID;

model.ChatModel.ChatTo = (int)model.NoteModel.UserID;

model.ChatModel.ChatFlag = 0;

model.ChatModel.ChatTime = System.DateTime.Now;

db.Chat.Add(model.ChatModel);

db.SaveChanges();

return Content("<script >alert('已成功提交！');history.go(-1)</script >", "text/html");

}

**运行页面：**

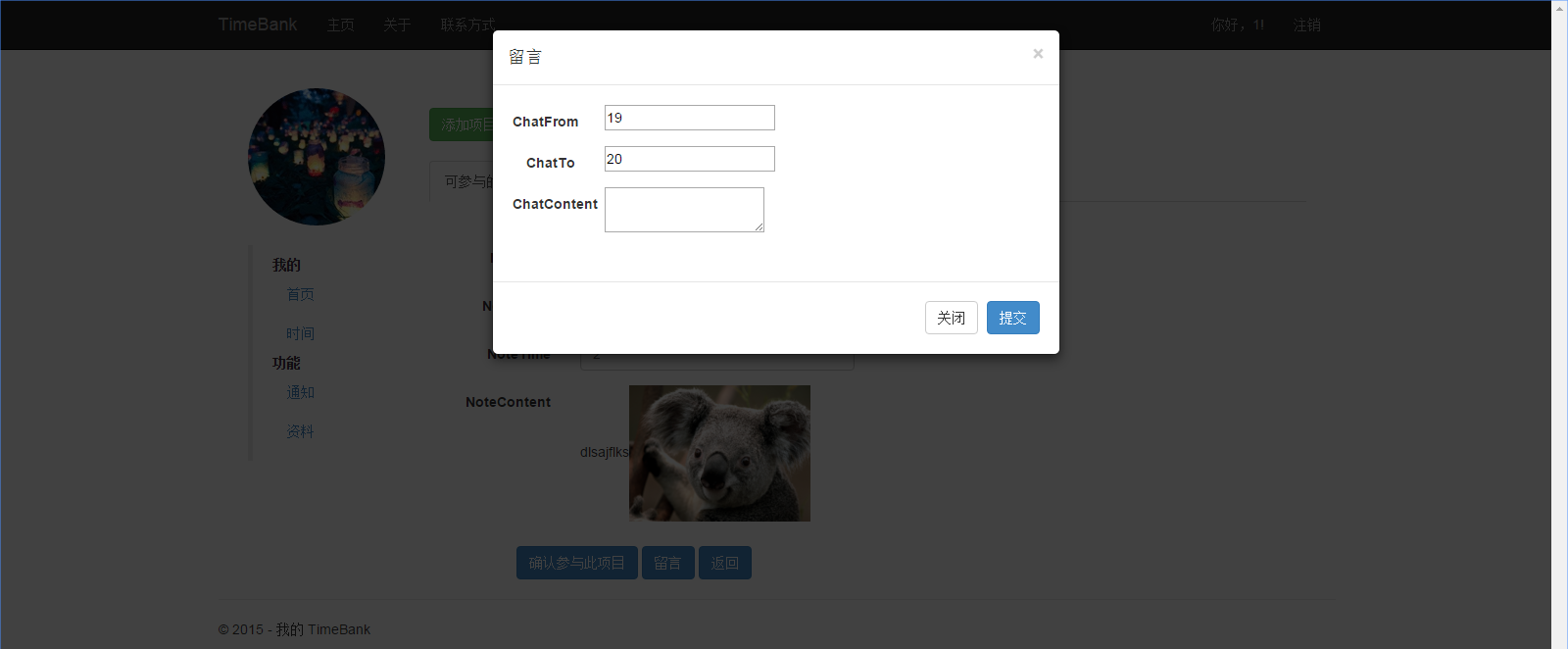


图5-22 项目内留言页面

### 5.16 用户通知

在通知页面上，用户可以查看所有消息信息，已读未读消息已经标出，若点击阅读，留言发送者也能查看到留言被已经阅读。

**代码：**

public ActionResult ChatsIndex()

{

String usercode = Session["username"].ToString();

int nid = 0;

foreach (var item in db.User.ToList<User>())

{

var Result = from a in db.User

where a.UserCode == usercode

select a;

foreach (var result in Result)

{

nid = result.UserID;

}

}

User user = db.User.Find(nid);

var chat = db.Chat.Include(c => c.User).Include(c => c.User1);

ViewModel model = new ViewModel();

model.UserModel = user;

model.ChatModel = chat.ToList();

return View(model);

}

**运行页面：**

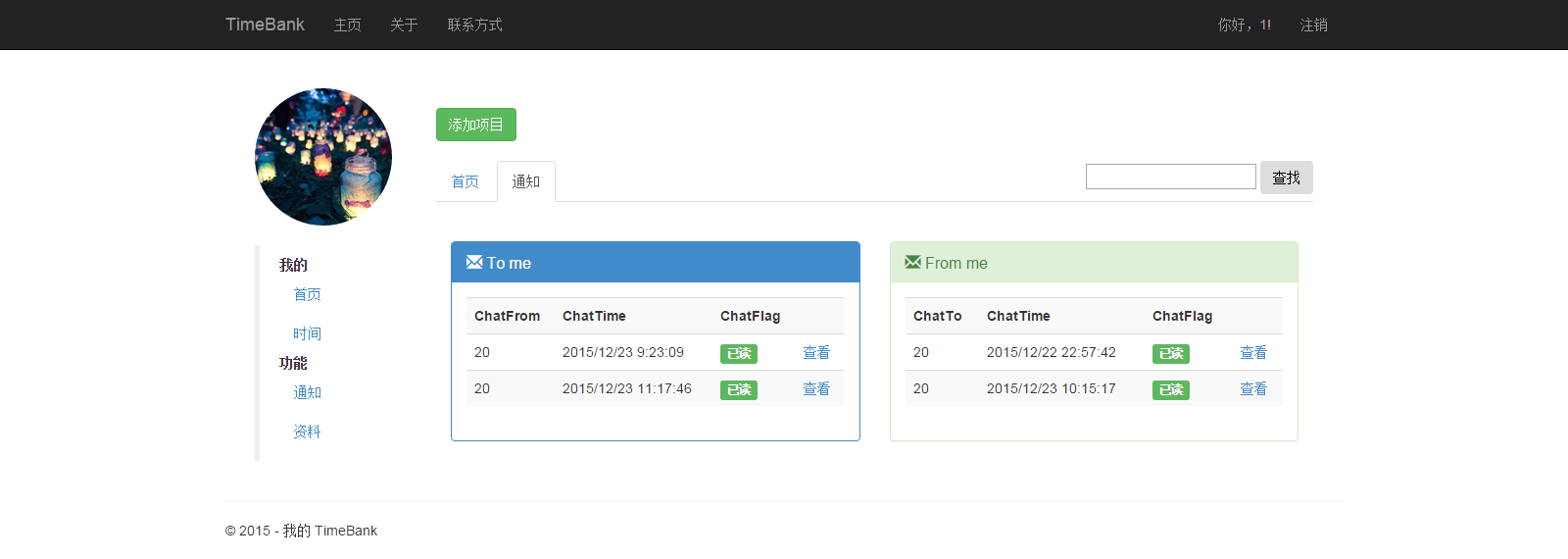


图5-23 用户通知页面

## 测试

在程序完成后，进行程序调试是很重要的。好的程序是要对于键盘上输入的任何值都不会有出错，而测试的目的就是尽可能的发现程序中的错误，要考虑到尽可能多的情况。

### 6.1 测试

* 情况一：用户登录错误提醒

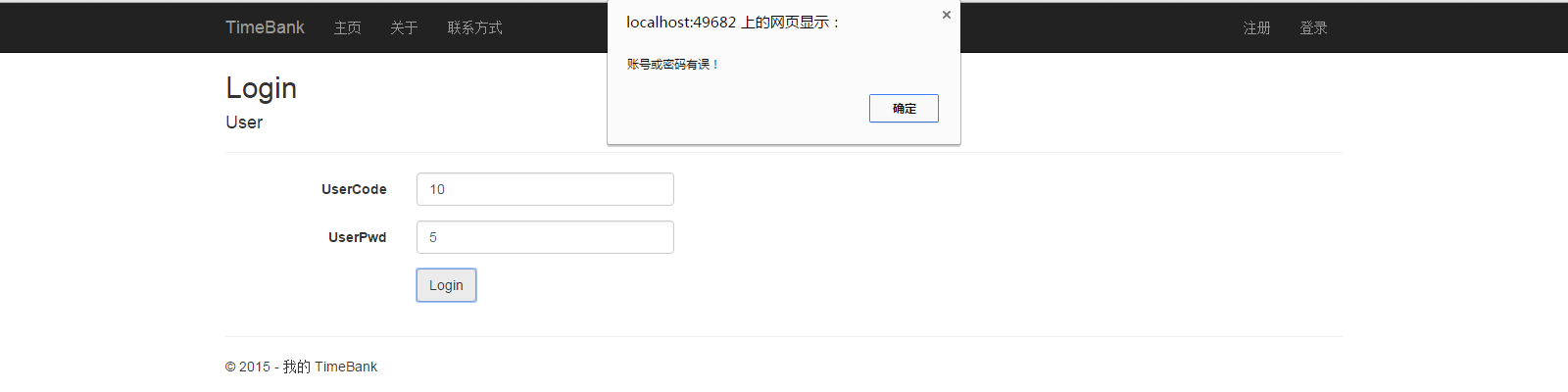


图6-1 用户登录错误提醒

* 情况二：项目已经被参与，该项目是不能删除和修改的，只能查看

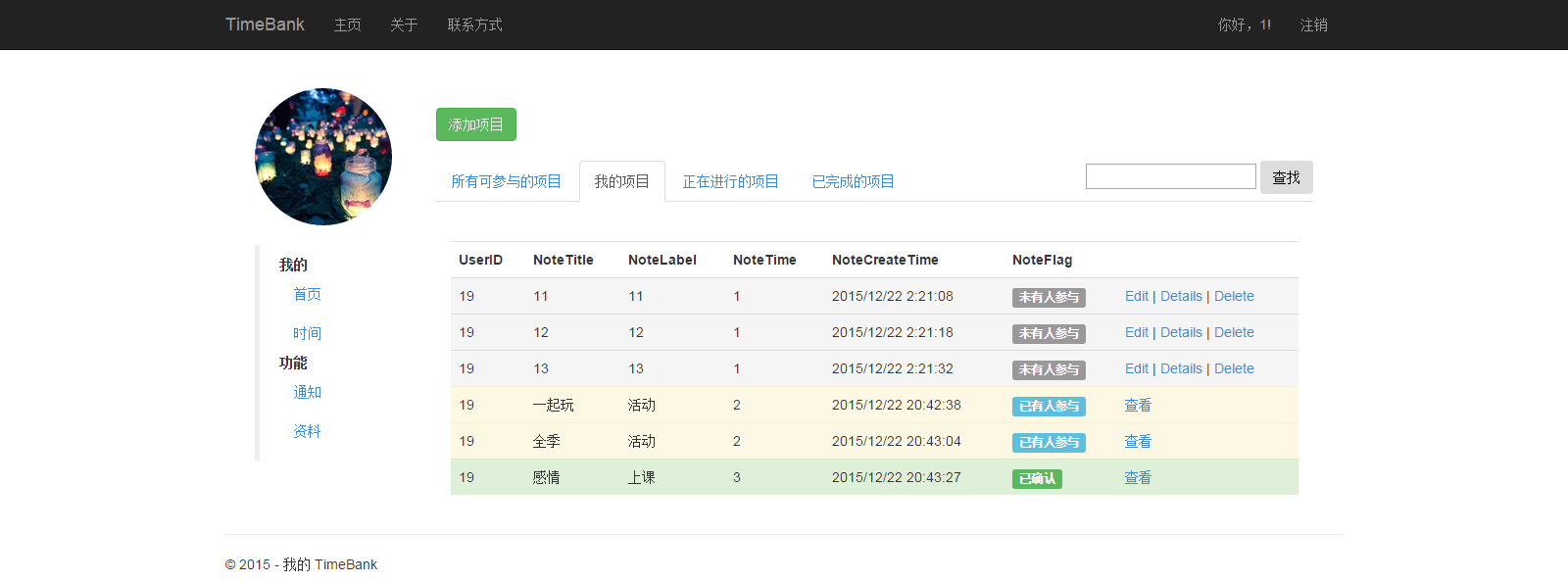


图6-2 项目情况判断

* 情况三：用户时间为零，不能再创建项目

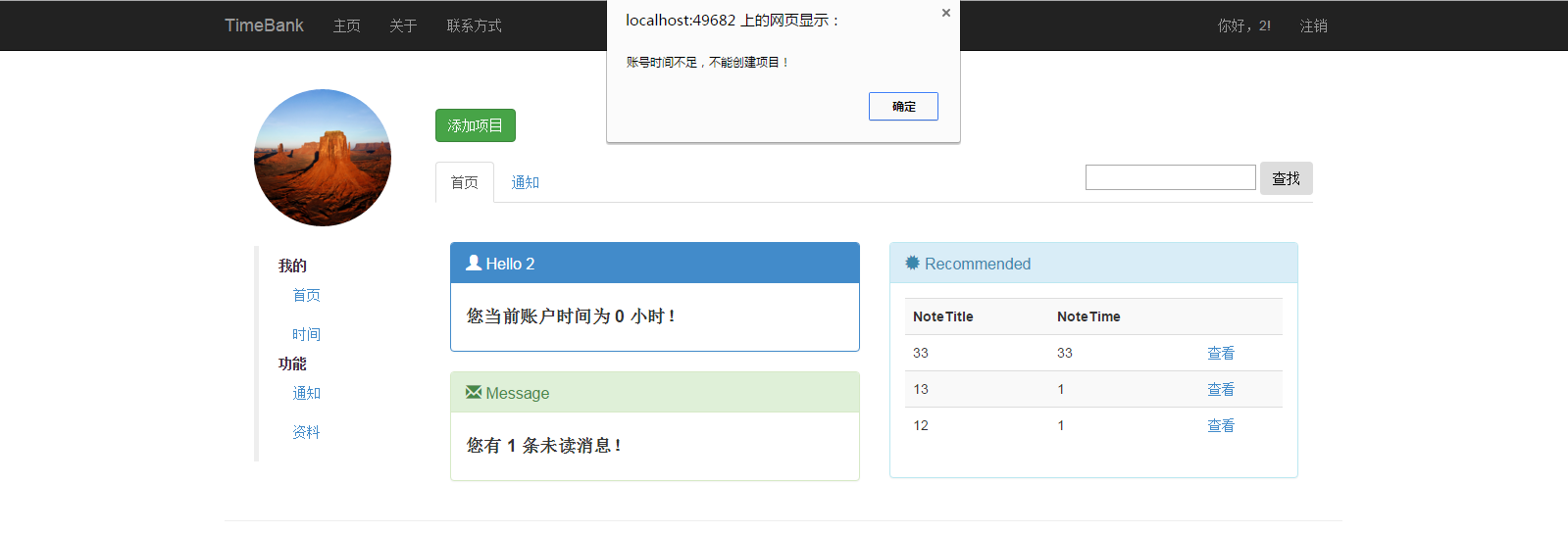


图6-3 账户余额不足

* 情况四：用户时间小于项目时间

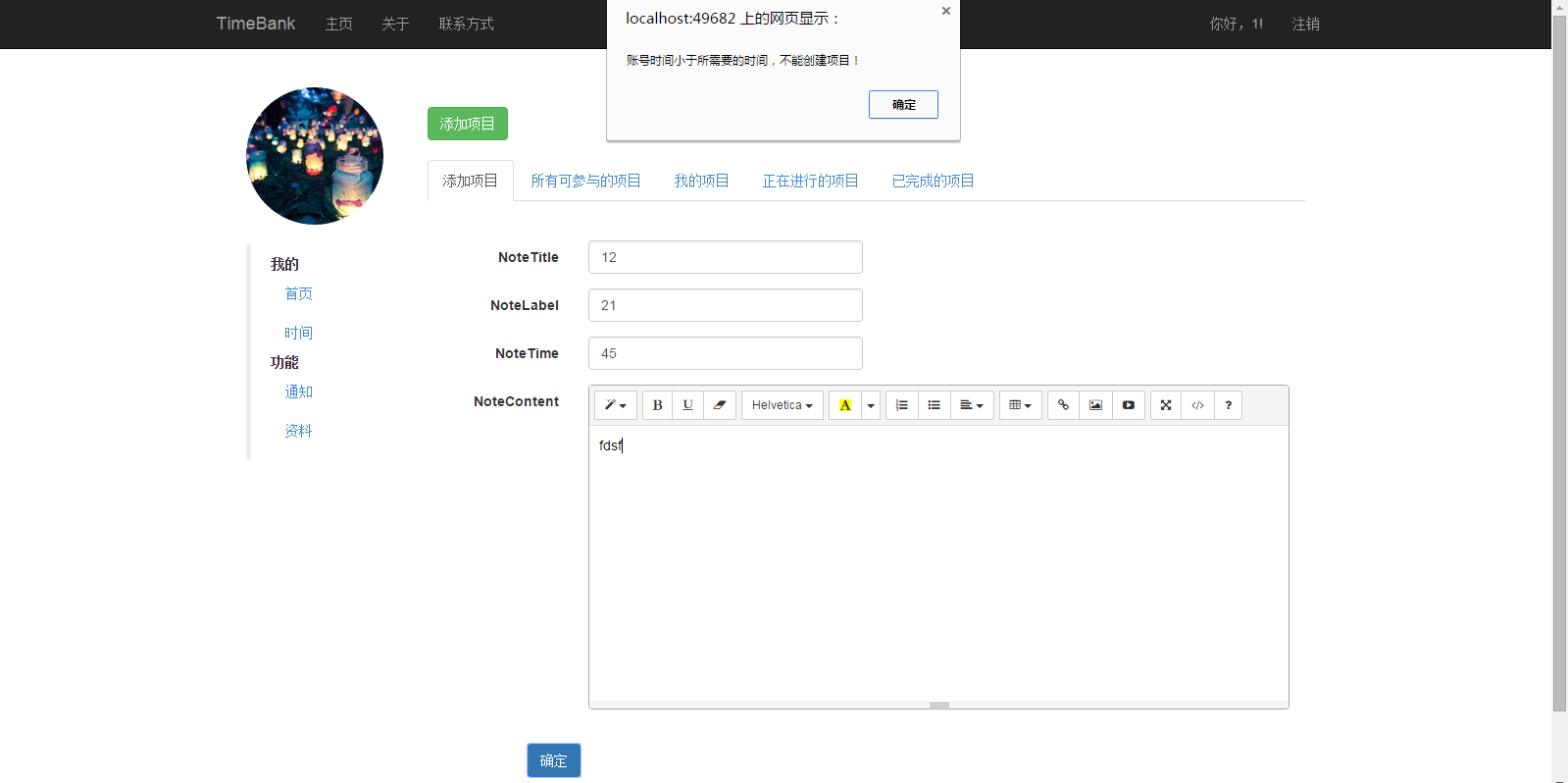


图6-4 用户时间小于所需时间

### 6.2 系统存在不足之处

系统在修改项目信息的时候图片呈现为二进制代码，不能显示，由于是利用Summornote富文本编辑器这个空间，还未找到相应的方法实现，待自己编写方法实现此功能

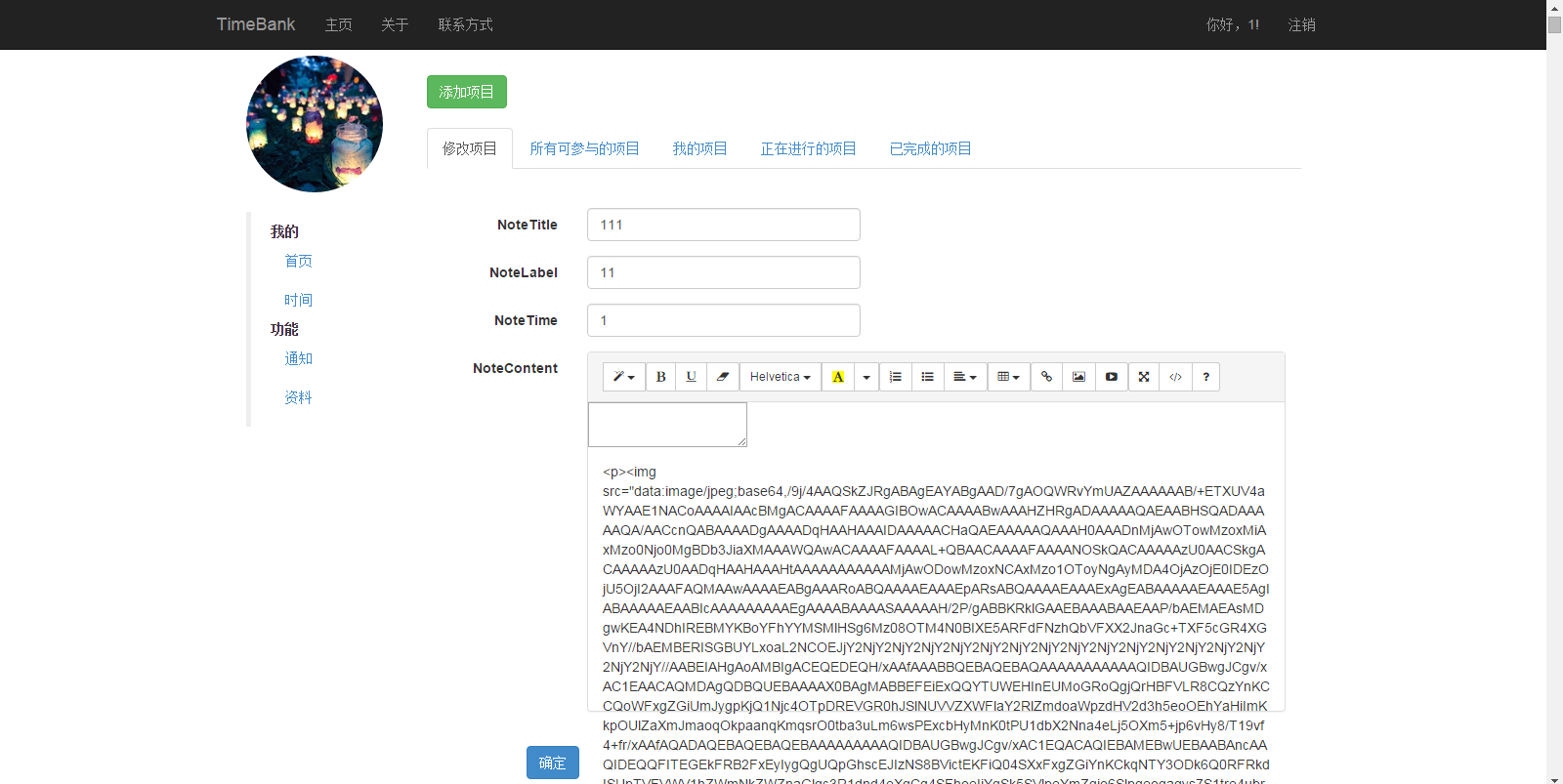


图6-5 图片呈现

## 七、总结

在本次课程设计中，MVC的框架结果帮助了我了解MVC体系的业务逻辑处理和页面呈现。采用CodeFirst建立Model类型和使用Linq语句查询数据库的方法都是之前没有用过的。在页面呈现上，我采用了Bootstrap的二次开发工具实现页面的呈现。

在业务逻辑上，我采用Session的方法来判断用户的登录状态，在每个页面上我都采用了Filter的方法来判断用户的权限。在页面视图要采用多种Model类型的时候，我使用了ViewModel类型来再次定义Model使得在页面上可以实现两种以上Model类型的内容。文本域的编写，采用Summornote控件实现文字、图片、连接、多媒体的编辑。用户头像的上传，我采用了Ajax的异步上传模式实现，并以二进制的方式存入数据库中。