Web 程序设计期末大作业：

图书管理系统

1. 业务功能简述

我设计了基于 ASP.NET MVC 的图书管理系统，数据库部分使用了 Entity Framework 的 Code First 模型，把用户分为普通用户（可借还书）和管理员用户（可添加书），实现了借书，还书，查询，添加书籍的功能。

2. 模型与数据库实体

我共设计了 Book, Admin ,Borrow, User 这四个类，代码如下：

Admin 类有 NameId 和 Pawword 这两个属性，Admin 类代表管理员。

```
7   namespace 图书管理系统.Models
8   {
        3 references
9       public class Admin
10      {
            1 reference
11          public Admin(string name, string Password)
12          {
13              this.NameId = name;
14              this.Password = Password;
15          }
            0 references
16          public Admin() { }
17          [Key]
            2 references
18          public string NameId { get; set; }
            2 references
19          public string Password { get; set; }
20      }
21  }
```

Book 类有 BookId , Title, Author, Total, Stock 这 5 个属性。

```
namespace 图书管理系统.Models
{
    10 references
    public class Book
    {
        [Key]
        2 references
        public string BookId { get; set; }
        2 references
        public string Title { get; set; }
        2 references
        public string Author { get; set; }
        1 reference
        public int Total { get; set; }
        4 references
        public int Stock { get; set; }
        3 references
        public Book(string BookId, string Title, string Author, int Total, int Stock)
        {
            this.BookId = BookId;
            this.Title = Title;
            this.Author = Author;
            this.Total = Total;
            this.Stock = Stock;
        }
        //定义无参数的构造函数主要是因为在通过DbSet获取对象进行linq查询时会报错
        //The class 'EFCodeFirstModels.Student' has no parameterless constructor.
        0 references
        public Book() { }
    }
}
```

User 类代表普通的可借书用户，有用户名和密码。

```
namespace 图书管理系统.Models
{
    3 references
    public class User
    {
        1 reference
        public User(string NameId, string password)
        {
            this.NameId = NameId;
            this.password = password;
        }
        0 references
        public User() { }
        [Key]
        2 references
        public string NameId { get; set; }
        2 references
        public string password { get; set; }
    }
}
```

Borrow 类代表借书信息，表的键由书 ID 和用户 ID 共同组成。

```
namespace 图书管理系统.Models
{
    3 references
    public class Borrow
    {
        1 reference
        public Borrow(string NameId, string bookId)
        {
            BookId = bookId;
            this.NameId = NameId;
        }
        0 references
        public Borrow() { }
        [Key]
        [Column(Order = 1)]
        3 references
        public string BookId { get; set; }
        [Key]
        [Column(Order = 2)]
        3 references
        public string NameId { get ; set ; }

    }
}
```

我使用的是 Code First，代码在 CodeFirst1.cs 里，代码如下：

```
4 references
public class CodeFirst1 : DbContext
{
    //您的上下文已配置为从您的应用程序的配置文件(App.config 或 Web.config)
    //使用“CodeFirst1”连接字符串。默认情况下，此连接字符串针对您的 LocalDb 实例上的
    //“图书管理系统.Models.CodeFirst1”数据库。
    //
    //如果您想要针对其他数据库和/或数据库提供程序，请在应用程序配置文件中修改“CodeFirst1”
    //连接字符串。
    1 reference
    public CodeFirst1()
        : base("name=CodeFirst1")
    {
        Database.SetInitializer<CodeFirst1>(new DropCreateDatabaseAlways<CodeFirst1>());
    //      Database.SetInitializer(new MigrateDatabaseToLatestVersion<CodeFirst1, Configuration>());
    //      Database.SetInitializer<CodeFirst1>(new MigrateDatabaseToLatestVersion<CodeFirst1, ReportingDbMigrationsConfiguration>());
    }
    8 references
    public DbSet<Book> Books { get; set; }
    2 references
    public DbSet<Admin> Admins { get; set; }
    2 references
    public DbSet<User> Users { get; set; }
    5 references
    public DbSet<Borrow> Borrows { get; set; }
    // public virtual DbSet<MyEntity> MyEntities { get; set; }
}

}
```

我从每个类派生了一个表。

本来数据的初始化应该放到 seed 函数里，我放到了 HomeController 的 inital_db 函数里，

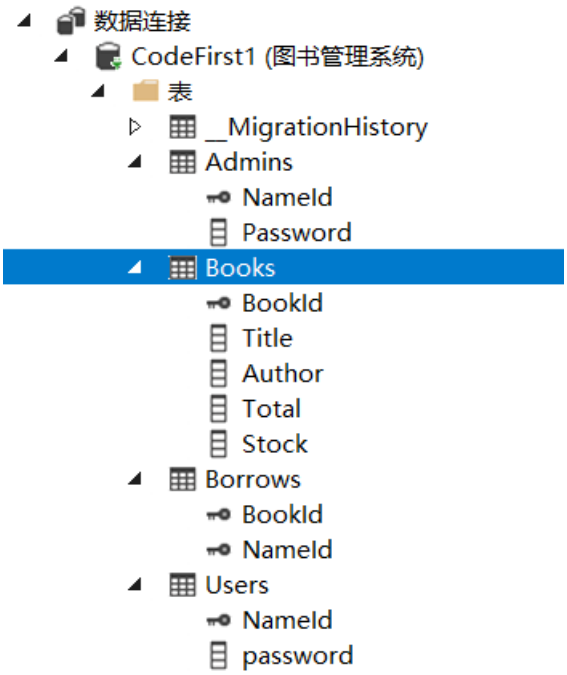
代码如下：

```
17      public int inital_flag = 0;
        21 references
18      public CodeFirst1 db = new CodeFirst1();
        4 references
19      public string Sha256(string plainText)
20      {
21          SHA256Managed _sha256 = new SHA256Managed();
22          byte[] _cipherText = _sha256.ComputeHash(Encoding.Default.GetBytes(plainText));
23          return Convert.ToBase64String(_cipherText);
24      }
25
        1 reference
26      public void inital_db()
27      {
28
29          var book1 = new Book("11111","C#图解教程", "索利斯",5,1);
30          var book2 = new Book("2222", "SQL进阶教程", "MICK",10,9);
31          var admin = new Admin("admin", Sha256( "admin"));
32          var user = new User("user", Sha256("user"));
33          db.Books.Add(book1);
34          db.Books.Add(book2);
35          db.Admins.Add(admin);
36          db.Users.Add(user);
37          db.SaveChanges();
38      }
```

程序会在载入主页面时对数据库进行初始化。

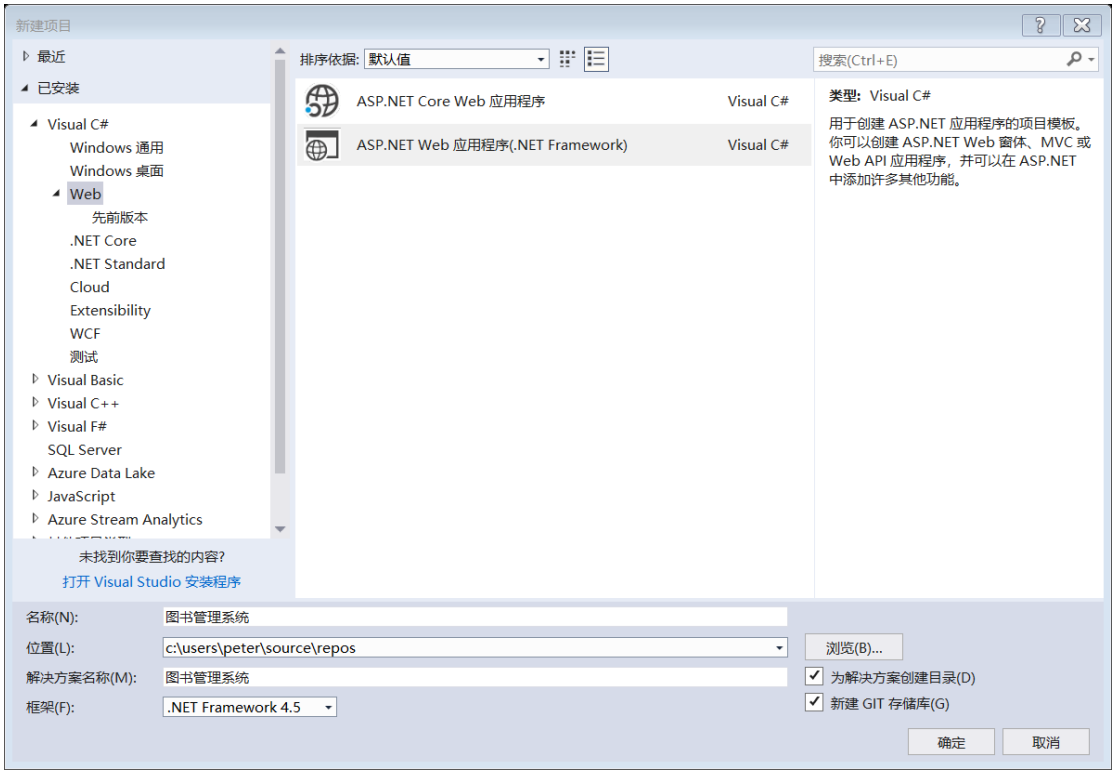
本来想用 MD5 存放加密后的密码，后来考虑到 MD5 不够安全，我改用了 SHA256.

数据库的实体模型如下：

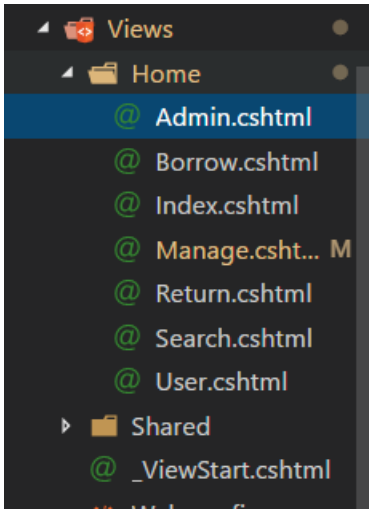


3. 网页功能与逻辑

我使用.NET4.5 建 MVC 框架。



我共设计了 7 个页面，如下：



我使用了_Layout 页面。

首先是 Index 页面和控制器：

```
<div class="jumbotron">
    <h1>图书管理系统</h1>
    <p class="lead">期末大作业：图书管理系统</p>
    <p><a href="" class="btn btn-primary btn-lg">Learn more &raquo;</a></p>
</div>
```

```
0 references
public ActionResult Index()
{
    if (inital_flag == 0)
    {
        inital_flag = 1;
        inital_db();
    }
    return View();
}
```

然后是 User 页面，实现用户登录功能：

```
0 references
public ActionResult User()
{
    var name = Request["name"];
    var password = Request["password"];
    if (name != null && password != null)
    {
        ViewBag.Message = UserLogin(name, password);
    }
    else
    {
        ViewBag.Message = -1;
    }
    return View();
}
}
```

其中 UserLogin 函数判断用户是否在数据库中：

```
1 reference
public int UserLogin(string  id, string password)
{
    int flag = 0;
    string Sha256_password = Sha256(password);
    var query = from User in db.Users
                where User.NameId == id && User.password == Sha256_password
                select User;
    foreach (var i in query)
    {
        flag = 1;
        Session["User"] = id;
    }
    return flag;
}
```

然后 Admin 页面实现管理员登陆功能：

```
0 references
public ActionResult Admin()
{
    var name = Request["name"];
    var password = Request["password"];
    if (name != null && password != null)
    {
        ViewBag.Message = AdminLogin(name,password);
    }
    else
    {
        ViewBag.Message = -1;
    }

    return View();
}
```

AdminLogin 函数判断管理员用户是否在数据库中

```
1 reference
public int AdminLogin(string name,string password)
{
    int flag = 0;
    string Sha256_password = Sha256(password);
    var query = from admin in db.Admins
                where admin.NameId==name && admin.Password== Sha256_password
                select admin;
    foreach(var  i in query)
    {
        flag = 1;
        Session["Admin"] = name;
    }
    return flag;
}
```

Borrow 页面进行借书：

```
0 references
public ActionResult Borrow()
{
    ViewBag.Message = JudgeUser();
    var BookId = Request["BookId"];
    if (BookId != null)
    {
        ViewBag.success = Book_Borrow(BookId);
    }
    else
    {
        ViewBag.success = -1;
    }
    return View();
}
```

其中 JudgeUser 函数判断用户是否登陆：

```
2 references
public int JudgeUser()
{

    if (Session["User"] != null)
    {
        return 1;
    }
    return 0;
}
```

Book_Borrow 函数进行借书：

```
1 reference
public int Book_Borrow(string BookId)
{
    int flag = 0;
    int stock = -1;
    var query = from Book in db.Books
                where Book.BookId == BookId
                select Book;
    foreach (var i in query)
    {
        flag = 1;
        stock = i.Stock;
    }
    if (flag == 0)
        return 0;//不存在这本书
    if (stock <= 0)
        return -2;//没库存
    if (Session["User"] == null)
    {
        return -4;//未登录，强行post
    }
    string user = Session["User"].ToString();
    var query2 = from Borrow in db.Borrows
                 where Borrow.BookId == BookId && Borrow.NameId== user
                  select Borrow;
    foreach (var i in query2)
    {
        return -3;//已借过这本书
    }
    var borrow = new Borrow(Session["User"].ToString(), BookId);
    db.Borrows.Add(borrow);
    var book = db.Books.Find(BookId);
    book.Stock -= 1;
    db.SaveChanges();
return 1;
}
```

Return 页面还书：

```
0 references
public ActionResult Return()
{
    ViewBag.Message = JudgeUser();
    var BookId = Request["BookId"];
    if (BookId != null)
    {
        ViewBag.success = Book_Return(BookId);
    }
    else
    {
        ViewBag.success = -1;
    }
    return View();
}
```

Book_Return 函数进行还书操作：

```
1 reference
public int Book_Return(string BookId)
{
    string user = Session["User"].ToString();
    if (Session["User"] == null)
    {
        return -4;//未登录，强行post
    }
    int flag = 0;
    var query = from Borrow in db.Borrows
                where Borrow.BookId == BookId && Borrow.NameId== user
                select Borrow;
    foreach (var i in query)
    {
        flag = 1;
    }
    if (flag == 0)
    {
        return 0;//没借这本书
    }
    //    var borrow = new Borrow(Session["User"].ToString(), BookId);
    var borrow = db.Borrows.Find( BookId, user);
    db.Borrows.Remove(borrow);
    var book = db.Books.Find(BookId);
    book.Stock += 1;
    db.SaveChanges();
    return 1;
}
```

Manage 页面主要用于管理员添加书籍：

```
0 references
public ActionResult Manage()
{
    ViewBag.Message = JudgeAdmin();
    var BookId = Request["BookId"];
    var Title = Request["Title"];
    var Author = Request["Author"];
    var Total = Request["Total"];
    if (BookId != null && Title != null && Author != null && Total != null)
    {

        ViewBag.success = AddBook(BookId, Title, Author, int.Parse(Total));
    }
    else if (BookId == null && Title == null && Author == null && Total == null)
    {
        ViewBag.success = -1;
    }
    else
    {
        ViewBag.success = -2;
    }
    return View();
}
```

AddBook 函数用来添加书：

```
1 reference
public int AddBook(string BookId, string Title, string Author, int Total)
{
    if (Total <= 0)
    {
        return -3;//书的数量不能小于0
    }
    var book = db.Books.Find(BookId);
    if (book != null)
    {
        return -4;//不能添加已在数据库中的书
    }
    var book1 = new Book(BookId, Title, Author, Total, Total);
    db.Books.Add(book1);
    db.SaveChanges();
    return 1;
}
```

最后的 Search 页面用来查找书籍：

```
269     public ActionResult Search()
270     {
271         var Title = Request["Title"];
272         var Author = Request["Author"];
273         if (Title != null || Author != null)
274         {
275             ViewBag.result = Book_Search(Title, Author);
276             ViewBag.Message = 0;
277         }
278         else
279         {
280             List<Book> books = new List<Book>();
281             ViewBag.result = books;
282             ViewBag.Message = -1;
283         }
284         return View();
285     }
```

Book_Search 函数用于查找书籍：

```
1 reference
91      public List<Book> Book_Search(string Title,string Author)
92      {
93          int flag = 0;
94          List<Book> books = new List<Book>();
95          var query = from Book in db.Books
96                      where Book.Title.Contains(Title)  && Book.Author.Contains(Author)
97                      select Book;
98          foreach (var i in query)
99          {
100             flag = 1;
101             books.Add(i);
102         }
103         return books;
104     }
```

函数返回的是书籍的列表。

Search.cshtml：

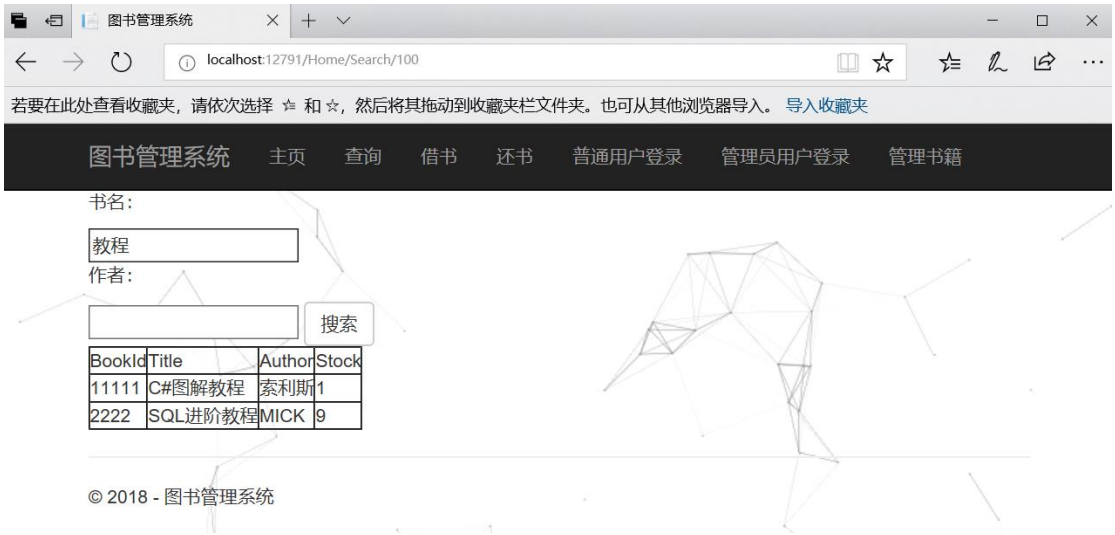
```
@using (Html.BeginRouteForm("Default", new { controller = "Home", action = "Search", Id = 100 }, FormMethod.Post))
{
    <p>书名: </p> <input type="text" name="Title" />
    <p>作者: </p><input type="text" name="Author" />
    <input type="submit" value="搜索" class="btn btn-default" />
}
    <table border="1">
        <tr>
            <td>BookId</td>
            <td>Title</td>
            <td>Author</td>
            <td>Stock</td>
        </tr>
        @{
            foreach (var i in ViewBag.result)
            {
                <tr>
                    <td>@i.BookId</td>
                    <td>@i.Title</td>
                    <td>@i.Author</td>
                    <td>@i.Stock</td>
                </tr>
            }
        }
    </table>
```

4. 操作演示

Index 页面：

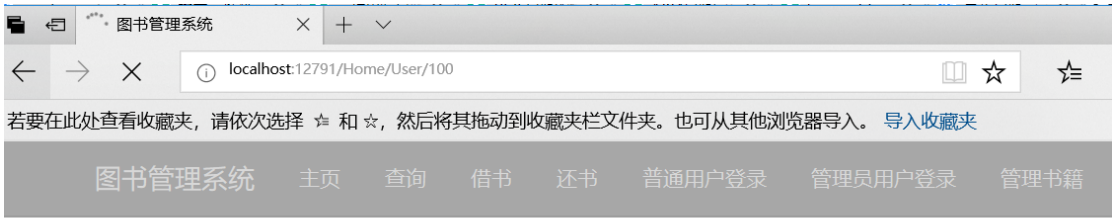
可以进行查询：

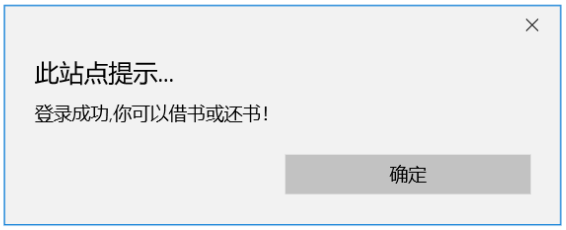
搜索带教程这 2 个字的书：



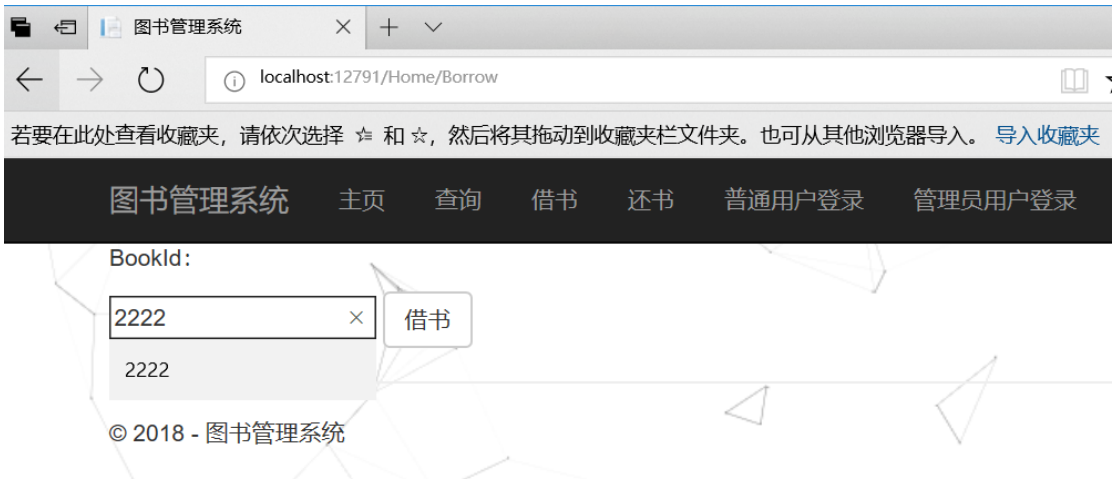
普通用户登录：

登录成功：

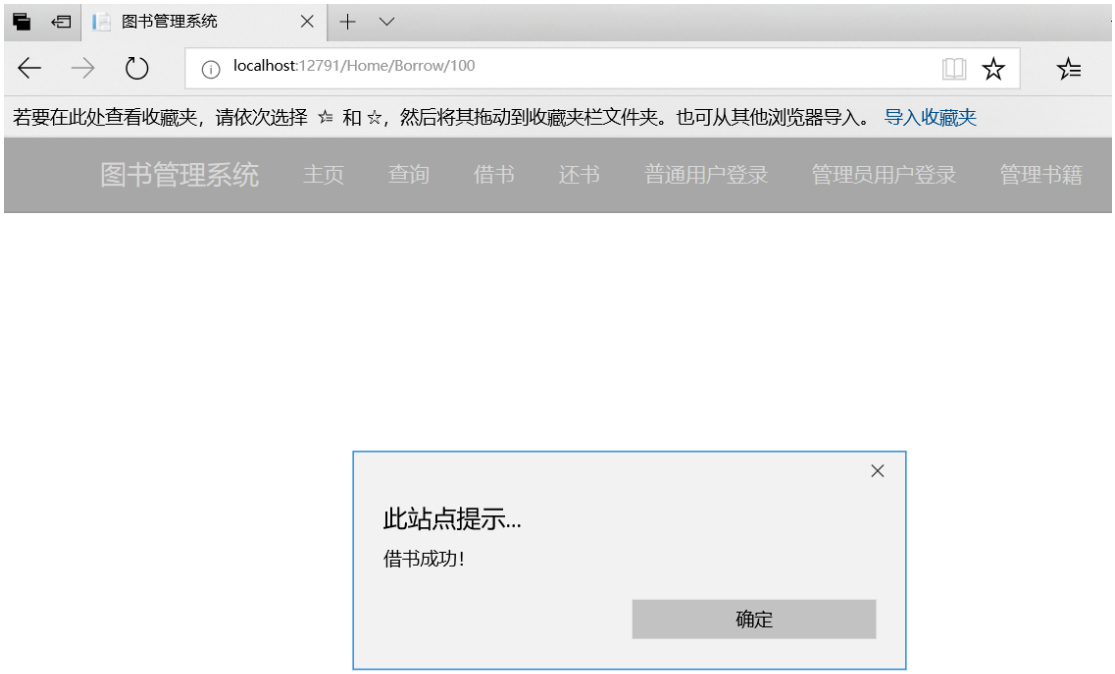




借书号为 2222 的书：



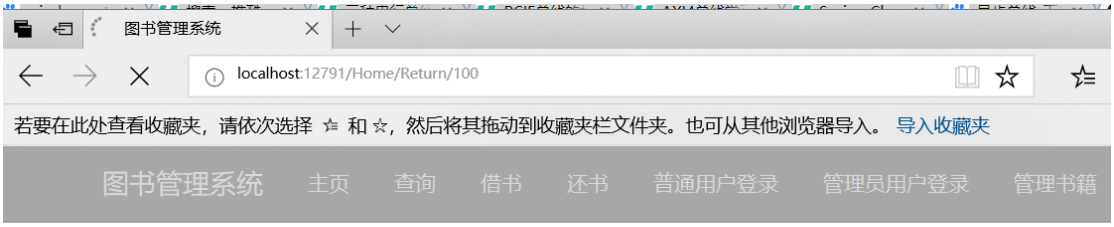
借书成功：

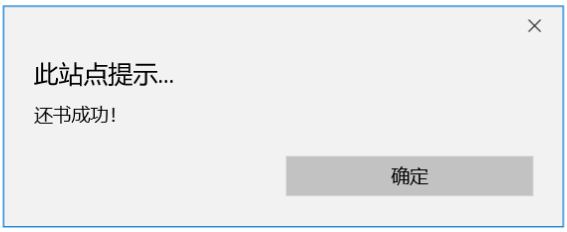


如果再借同一本书，会显示已经借过了：

然后还书：

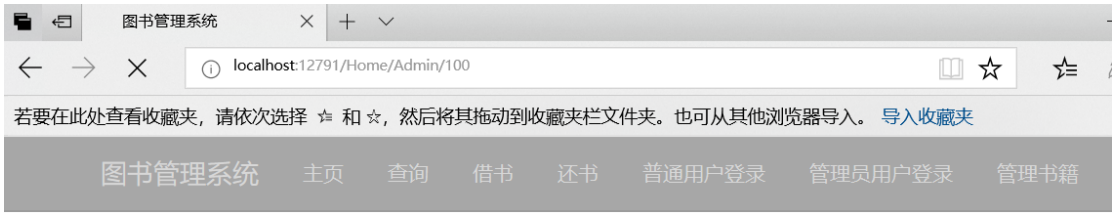

图书管理系统
localhost:12791/Home/Return
若要在此处查看收藏夹，请依次选择 ☆ 和 ☆，然后将其拖动到收藏夹栏文件夹。也可从其他浏览器导入。 导入收藏夹
图书管理系统 主页 查询 借书 还书 普通用户登录 管理员用户登录 管理书籍
BookId:
2222
还书
© 2018 - 图书管理系统

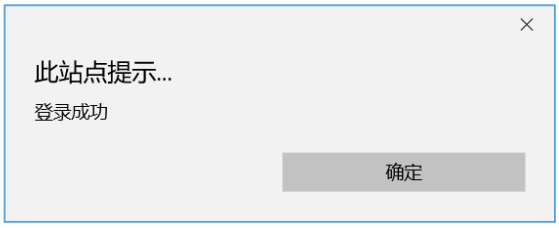

图书管理系统
localhost:12791/Home/Return/100
若要在此处查看收藏夹，请依次选择 ☆ 和 ☆，然后将其拖动到收藏夹栏文件夹。也可从其他浏览器导入。 导入收藏夹
图书管理系统 主页 查询 借书 还书 普通用户登录 管理员用户登录 管理书籍


此站点提示...
还书成功!
确定

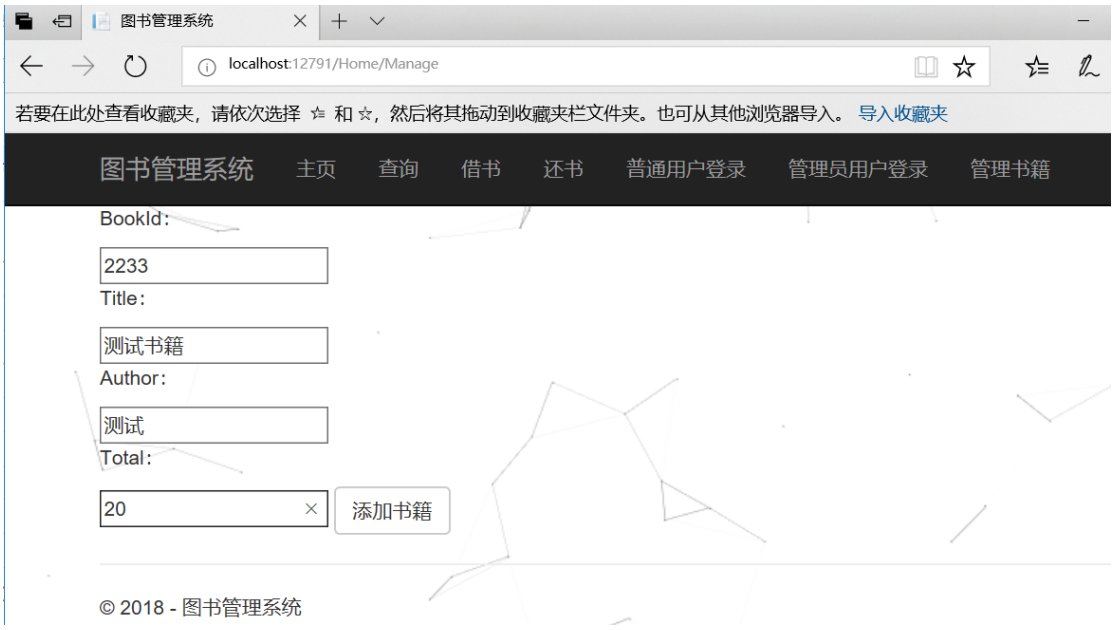
Admin 登录：


图书管理系统
localhost:12791/Home/Admin
若要在此处查看收藏夹，请依次选择 ☆ 和 ☆，然后将其拖动到收藏夹栏文件夹。也可从其他浏览器导入。 导入收藏夹
图书管理系统 主页 查询 借书 还书 普通用户登录 管理员用户登录 管理书籍
名字:
admin
密码:
admin
管理员登录
© 2018 - 图书管理系统


图书管理系统
localhost:12791/Home/Admin/100
若要在此处查看收藏夹，请依次选择 ☆≡ 和 ☆，然后将其拖动到收藏夹栏文件夹。也可从其他浏览器导入。 导入收藏夹
图书管理系统 主页 查询 借书 还书 普通用户登录 管理员用户登录 管理书籍


此站点提示...
登录成功
确定

添加书籍：


图书管理系统
localhost:12791/Home/Manage
若要在此处查看收藏夹，请依次选择 ☆≡ 和 ☆，然后将其拖动到收藏夹栏文件夹。也可从其他浏览器导入。 导入收藏夹
图书管理系统 主页 查询 借书 还书 普通用户登录 管理员用户登录 管理书籍
BookId:
2233
Title:
测试书籍
Author:
测试
Total:
20
添加书籍
© 2018 - 图书管理系统

页面说明完毕。

5. 安装与使用说明

我使用的.NET 框架是.NET4.5 版本，使用的数据库是 sql server。

因为我用的是 Code First，所以实际上并不需要初始化数据库，但为了使用方便，我生成了一个需要初始化数据的 publish 版本和不需要初始化数据的版本，这 2 个版本的使用都需要修改 WebConfig 里的连接参数。