國立嘉義大學資訊工程學系
資料庫系統設計期末專題報告

簡易銀行系統

學生： 1072948 翁陽龍

指導教授： 陳耀輝

中華民國 一百一十 年 一 月

摘要

本專題的目的在於透過 MVC 架構以及資料庫系統製作出具有新增帳號、登入、登出、提款、存款、轉帳及交易紀錄查詢等功能的簡易版銀行系統。此外，為了增加系統安全性，儲存密碼時亦使用了雜湊加鹽的技術。

關鍵詞：MVC 架構、資料庫

目錄

一、系統架構介紹

1.1 系統概述

MVC 架構顧名思義分為三個部分：模型、檢視、控制器。其中模型負責資料的處理及定義；檢視負責將結果輸出至網頁上；而控制器負責模型及檢視之間的協調。以下將分為三個小節介紹這三個部分較細節的情況。

1.2 模型

模型的部分包含一個資料庫實體模型做為資料來源、四個表單定義和一個表格定義做為和檢視搭配的資料定義(見圖 1.1)。資料庫綱要分為兩個資料表，Account 及 Transaction_log，主鍵分別為 username 和 tid，後者有兩個外來鍵 name 和 destination ，都對應到 username。由於 destination 是轉帳時才會用到的欄位，設為允許為空。由於每個資料表都只有單一主鍵，而且每個欄位都沒有遞移相依的情況，這個資料庫綱要符合 BCNF 正規化(見圖 1.2)。

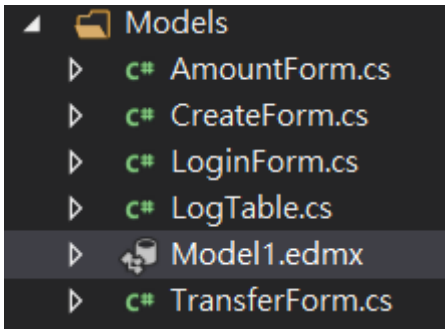

圖 1.1: 模型

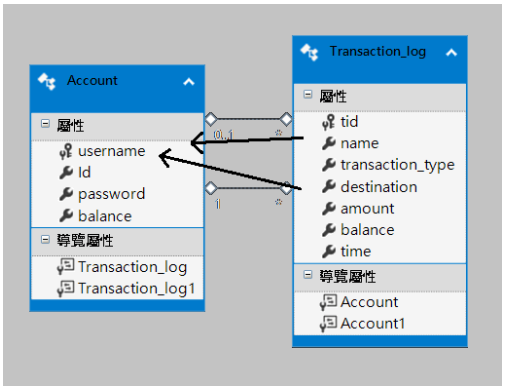

圖 1.2: 資料庫綱要

```
public class LogTable
{
    [Key]
    1 個參考
    public int Tid { get; set; }
    [Display(Name = "交易類型")]
    3 個參考
    public string Type { get; set; }
    [Display(Name = "轉帳目標")]
    3 個參考
    public string Dest { get; set; }
    [Display(Name = "交易金額")]
    3 個參考
    public long Amount { get; set; }
    [Display(Name = "結餘")]
    3 個參考
    public long Balance { get; set; }
    [Display(Name = "交易時間")]
    3 個參考
    public string Time { get; set; }
}
```

圖 1.3: 資料定義範例

1.3 控制器

控制器中包含多個方法，其中大部分的方法都會對應到一個檢視，以在網頁上輸出內容。若有方法需要接收使用者輸入的資料，其方法需要分為 HTTP GET 和 HTTP POST 兩種，使得第一次載入網頁時不會接收空白的資料。另外，本專案使用 Linq 作為控制器和資料庫溝通的媒介。由於方法眾多，無法一一列出介紹，只能提供少數範例。

```
public ActionResult Create(CreateForm form)...
[Authorize]
0 個參考
public ActionResult Balance()...
[Authorize]
0 個參考
public ActionResult Withdraw()...
[HttpPost]
[Authorize]
0 個參考
public ActionResult Withdraw(AmountForm form)...
[Authorize]
0 個參考
public ActionResult Deposit()...
[HttpPost]
[Authorize]
0 個參考
public ActionResult Deposit(AmountForm form)...
[Authorize]
0 個參考
public ActionResult Transfer()...
[HttpPost]
[Authorize]
0 個參考
public ActionResult Transfer(TransferForm form)...
[Authorize]
0 個參考
public ActionResult Record()...
5 個參考
public void logGen(String name, byte type, String dest, long amount, long balance)...
```

class System.Web.Mvc.ActionResult
表示動作方法的結果。

圖 1.4: 方法列表(部分)

```
public ActionResult Record()
{
    String user = Session["account"] as string;
    List<LogTable> list = new List<LogTable>();
    var r = (from a in db.Transaction_log
             where a.name == user
             select a);
    foreach(var log in r)
    {
        byte t = log.transaction_type;
        string tType = "";
        switch (t)
        {
            case 1:tType = "提款";
                break;
            case 2:tType = "存款";
                break;
            case 3:tType = "轉出";
                break;
            case 4:tType = "轉入";
                break;
        }
        list.Add(new LogTable { Tid = log.tid, Type = tType, Dest
    }
    return View(list);
}
```

圖 1.5: Record 方法

```
[HttpPost]
[Authorize]
0 個參考
public ActionResult Transfer(TransferForm form)
{
    String user = Session["account"] as string;
    var r = (from a in db.Account
             where a.username == user
             select a).FirstOrDefault();
    var dest = (from a in db.Account
                where a.username == form.transferName
                select a).FirstOrDefault();
    if (r.balance < form.transferAmount) ViewBag.message = "餘額不足!";
    else if (dest == null) ViewBag.message = "此帳號不存在!";
    else if (user == form.transferName) ViewBag.message = "不能轉帳給自己!";
    else
    {
        r.balance = r.balance - form.transferAmount;
        logGen(user, 3, dest.username, form.transferAmount, r.balance);
        dest.balance = dest.balance + form.transferAmount;
        logGen(dest.username, 4, user, form.transferAmount, dest.balance);
        ViewBag.message = "操作成功!";
    }

    return View();
}
```

圖 1.6: Transfer 方法

1.4 檢視

檢視透過 HTML 以及 Razor 語法將查詢結果經過排版輸出在網頁上。由於網頁和資料庫關聯較少，在此僅提供一個範例，不多加贅述。

```
@{
    ViewBag.Title = "Create";
}

<h2>新增帳戶</h2>

@using (Html.BeginForm())
{
    <div>
        @Html.Label("帳號:")
        <input type="text" name="createUsername" required />
        <br />
        @Html.Label("密碼:")
        <input type="password" name="createPassword" required />
        <br />
        @Html.Label("存入金額:")
        <input type="number" min="0" name="createBalance" required />
        <br />
    </div>
    @ViewBag.message
    <input type="submit" value="新增" />
}
```

圖 1.7: Create 檢視

二、成果展示時的失誤及檢討

在課堂成果展示前兩個小時，我本來只剩下交易紀錄查詢功能尚未實作，卻發現資料庫綱要存在拼字錯誤。為了修復這個問題，我必須重新開啟一個專案，把錯誤或不適合的名稱改掉，把舊的程式碼搬運過去。因此我把剩下的時間都用掉了，也沒有足夠的時間去除錯以及充實並檢查簡報內容，造成成果展示時轉帳的部分出現問題(見圖 2.1)以及簡報內容有誤，後來繼續除錯才發現其實這些錯誤都不需要花太多時間就能找出來。這件事告訴了我資料庫綱要在建立時最好要一步到位，否則很有可能會因為改變綱要而額外付出大量的時間成本。

```
public void logGen(String name, byte type, String dest, long amount, long balance)
{
    Transaction_log log = new Transaction_log();
    var c = (from a in db.Transaction_log
             select a);
    log.tid = c.Count();
    log.name = name;
    log.transaction_type = type;
    if (dest != "") log.destination = dest;
    log.amount = amount;
    log.balance = balance;
    log.time = DateTime.Now;
    db.Transaction_log.Add(log);
    db.SaveChanges();//必須馬上更新log，tid才不會重複，課堂demo轉帳失敗的原因
```

圖 2.1: 造成轉帳功能失敗的 logGen 方法

三、專題成果

雖然沒能在課堂展示時完成所有功能，但是經過補全現在系統已達成所有預計要完成的功能了，運作時也完全正常。以下補充課堂上沒有展示到的部分。

圖 3.1: 轉帳正常運作

dbo.Transaction_log [資料] | Record.cshtml | LogTable.cs | AccountsController.cs | Transfer.cshtml | TransferF

最大資料列(O): 1000

	tid	name	transaction_ty...	destination	amount	balance	time
▷	0	123	2	NULL	2000	2000	2021/1/13 上午 04:54:45
	1	456	2	NULL	100	100	2021/1/13 上午 04:54:55
	2	123	3	456	500	1500	2021/1/13 上午 04:56:00
	3	456	4	123	500	600	2021/1/13 上午 04:56:00
	4	789	2	NULL	30000	30000	2021/1/13 上午 04:59:00
	5	456	3	789	300	300	2021/1/13 上午 04:59:20
	6	789	4	456	300	30300	2021/1/13 上午 04:59:20
	7	456	2	NULL	3000	3300	2021/1/13 上午 04:59:26
	8	456	1	NULL	1000	2300	2021/1/13 上午 04:59:56
	9	789	1	NULL	2000	28300	2021/1/13 上午 05:00:24
	10	789	3	123	25000	3300	2021/1/13 上午 05:00:50
	11	123	4	789	25000	26500	2021/1/13 上午 05:00:50
	12	456	3	123	500	1800	2021/1/13 上午 05:03:07
	13	123	4	456	500	27000	2021/1/13 上午 05:03:08
	14	789	3	456	3000	300	2021/1/13 上午 09:33:47
	15	456	4	789	3000	4800	2021/1/13 上午 09:33:47
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

圖 3.2: 從資料庫觀察交易紀錄

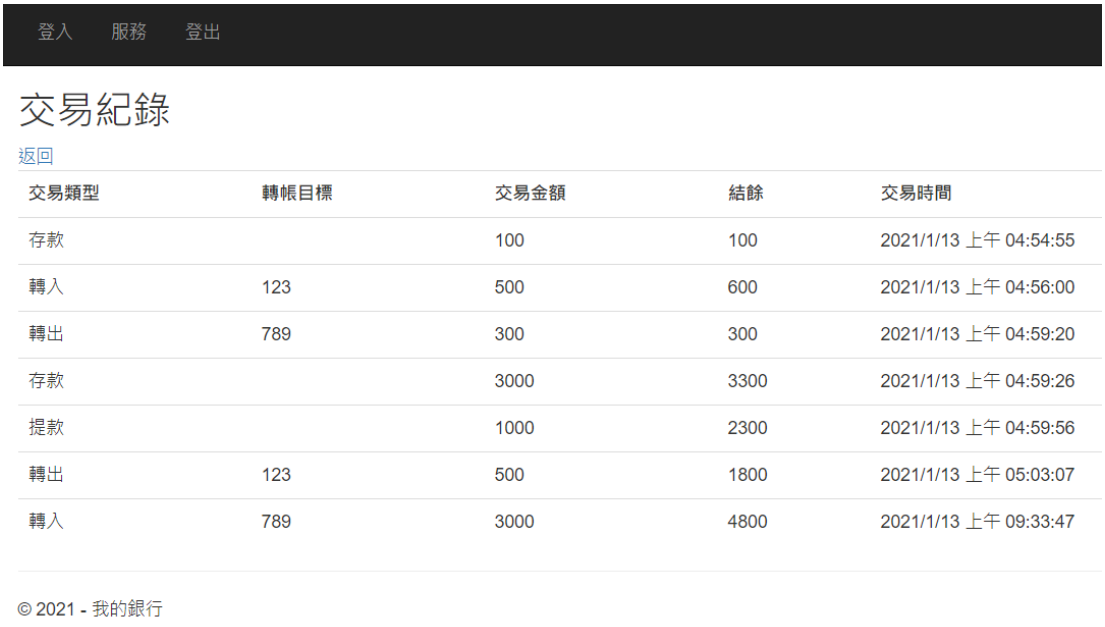

圖 3.3: 查詢 456 的交易紀錄(可與圖 3.2 對照)

四、結論

雖然獨自一人製作此專題很辛苦，但是有了這次的經驗，我發覺比起課堂上的講述及實作練習，製作專題對資料庫系統設計的了解更有幫助，也最能增加資料庫的實務經驗。