Vicious & Delicious (IFB299 Team 85)

Texchange Textbook Sharing

Process Design Document for Borrowing, Returning and Rating

Andrew Cooper
4th October 2015

Process Design Document for Borrowing, Returning & Rating on Texchange

Introduction

Purpose

This document serves to present a solution to the Borrowing/Returning problem identified during the Texchange meeting on October September 24th. The process by which a books is confirmed as borrowed and/or returned is not as simple as originally thought, and is a problem that needs a complete solution presented upfront as opposed to incrementally working towards a solution.

Scope

This document assumes a working knowledge of all Texchange subsystems to date. It does not aim to address the many implementation-dependant bugs that could be introduced, but rather only addresses the high level user experience flow issues identified within the above-mentioned meeting.

Overview

A button on the Book Information view allows a user to request to borrow a book. These requests are presented on the MyAccount page, and include options to Accept or Decline the request.

Borrowing Process

View

There are 3 key view components which need to be built to facilitate user interaction with the process, the Borrow button and email Modal, the Notifications component, and the Requests section within the MyAccount view.

“Borrow” Button

A “Borrow” button should be added to the “Book Details” view for each book. When a user wishes to borrow a book they’ll be able to look at the details of the book, and then click this button to begin the borrowing process.

The button should be styled with the standard modified-Bootstrap CSS as per all other buttons in the Texchange application.

Additionally, the button should pop an email form, contained within a Bootstrap Modal, that allows the user to send an email to the owner of the book. This should be styled as per the other email Modals already built into the “Contact” page.

“Notifications”

Although we’ll email the user regarding any events that occur on the Texchange platform, we also need to show these notifications on the site. One option is to include a number next to their Name on the Navigation Bar. For example: “Andrew (1)” would indicate 1 notification. The brackets and number should not be displayed for 0 notifications.

This is a justified position for the notifications for two reasons. First, the Nav Bar is always visible, so users will be able to see notifications from anywhere within the application. Secondly, the Name link currently directs the user to their MyAccount page, which is where Borrow Requests and similar will be displayed.

“Requests” section within MyAccount

We’ll add a “Requests” section to top of MyAccount page. This should be a table, including an identifier for the request, followed by an “Accept” button and a “Decline” button (for each request in the list).

The table should be styled using the master CSS stylesheet, as well as the included Bootstrap styling. Ditto for the “Accept” and “Decline” buttons.

Database

In order to maintain the number of notifications that a user has pertaining to their account and the books that they own, a Notifications column should be added to the Users Table. This should be of type Integer with a default value of 0. This will require updating the table definition.

Updated Table Definition

The updated table definition should be as follows (new column in bold):

```
CREATE TABLE [dbo].[AspNetUsers] (
    [Id]                    NVARCHAR (128)     NOT NULL,
    [Email]                 NVARCHAR (256)     NULL,
    [EmailConfirmed]        BIT                NOT NULL,
    [PasswordHash]          NVARCHAR (MAX)     NULL,
    [SecurityStamp]         NVARCHAR (MAX)     NULL,
    [PhoneNumber]           NVARCHAR (MAX)     NULL,
    [PhoneNumberConfirmed]  BIT                NOT NULL,
    [TwoFactorEnabled]      BIT                NOT NULL,
    [LockoutEndDateUtc]     DATETIME           NULL,
    [LockoutEnabled]        BIT                NOT NULL,
    [AccessFailedCount]     INT                DEFAULT ((0)) NOT NULL,
    [UserName]              NVARCHAR (256)     NULL,
    [FirstName]             NVARCHAR (256)     NULL,
    [LastName]              NVARCHAR (256)     NULL,
    [ContactNumber]         NVARCHAR (15)      NULL,
    [ThumbsUp]              INT                DEFAULT ((0)) NOT NULL,
    [ThumbsDown]            INT                DEFAULT ((0)) NOT NULL,
    [Tokens]                INT                DEFAULT ((0)) NOT NULL,
    [Notifications]         INT                DEFAULT ((0)) NOT NULL,
    CONSTRAINT [PK_dbo.AspNetUsers] PRIMARY KEY CLUSTERED ([Id] ASC)
);

GO
CREATE UNIQUE NONCLUSTERED INDEX [UserNameIndex]
    ON [dbo].[AspNetUsers]([UserName] ASC);
```

Controller

Form Submit on Email Modal

When the submit button on the email modal invoked by the “Borrow” button on the book information view is pressed, three things may need to happen.

1. The email needs to be sent, which requires checking for valid data. The email should be sent to the email address for the owner of the book, and the “from address” should be set to the email address for the currently logged in user.

2. If the email is successfully sent, the Borrower column in the Books table needs to be updated for the relevant Book entry. It should be changed from NULL to the username of the borrower.

3. Again only if the email is successfully sent, the Notifications column in the AspUsers table should be incremented, for the record of the user who owns the book.

If the email is not sent successfully, append an appropriate error message and return.

Allow the View to know the Notifications count

Requests for a notifications count for a user will come from the view via a Razor command. This will enable the view to dynamically display the number of notifications a user has. The controller can handle these requests by providing a public class method (public static) which returns the count of notifications from the users table.

Populate the Requests Table

The requests table can be populated with data from the Books table. There is no need for an additional Requests table.

A query similar to the following will achieve this.

```
select table from books
where (owner == logged-in-username) && (Borrower != null) && (LoanConfirmed == false)
or similar
```

Accepting or Declining a Borrow Request

“Accept” should set LoanConfirmed in the Books table to true. Both “Accept” and “Decline” should remove the book from the Requests section and email both users, and decrement Notifications in the Users table. A controller method will be required to perform this action.

Returning

View

Two key components need to be added to the MyAccount View in order to facilitate returning books (and rating borrowers).

My Books (on loan)

First, we need to add a “My Books (on loan)” section to the top of MyAccount view. This will be a table, styled the same as the Requests table detailed earlier in this document. It will be populated with a list of all the books the current user owns that are being borrowed.

Additionally, a “Returned?” button needs to appear next to each entry in this table. This button should be formatted with the standard master CSS and Bootstrap stylesheets.

Secondly, a “Please Leave Ratings” section should also be added to the top of the MyAccount view. This will be another table, style consistently with the “My Books (on loan)” table. This table will be populated with a list of users whom they need to leave ratings for, and will include a “Rate” button beside each entry.

This “Rate” button will pop a form contained in a Bootstrap Modal, that allows the current user to rate the user whom they borrowed a book from. The form should allow for a binary rating (thumbs up or thumbs down) and contain a field for a comment. The form and the Modal will be styled consistently with other Modals on the texchange platform: using the master CSS and Bootstrap stylesheets.

Database

Ratings and Comments Table

A new table needs to be created to contain ratings and comments. This table will contain columns for:

- a unique identifier
- an identifier for the user who left the rating
- an identifier for the user whom the rating is for
- a comment regarding the rating
- the binary ratings (either -1, 1, or 0, where 0 indicates that a rating has not yet been left).

Controller

Returning a Book

When a user submits the Rating form after pressing the Returned? Button, a controller method will handle the returning of the book and the rating of the user. The following needs to be achieved by this method:

1. Set in books table: Borrower to null, LoanConfirmed to null.
2. Add new entry to Ratings table:
    a. Set FromUsername to logged in user.
    b. Set ToUsername to the user the rating is about
    c. Set Comment to the comment they left.
    d. Set Rating in Ratings table to 1 or -1.
3. Add another entry to Ratings table:
    a. Set FromUsername to user who needs to rate them.
    b. Set ToUsername to currently logged in user.
    c. Leave other fields blank - rating will be 0 and we'll know they need to be rated.
4. Increment Notifications, in users table, for the user who needs to leave a rating.

"Need to leave a rating" notification

Most of the requirements for this controller method are already achieved by the method relating to "Returning a Book". There are just key requirements for this controller method:

- The database entry that is already in the Ratings table (which will contain no comment and a rating of "0"), needs to be updated
- The relevant notification needs to be cleared