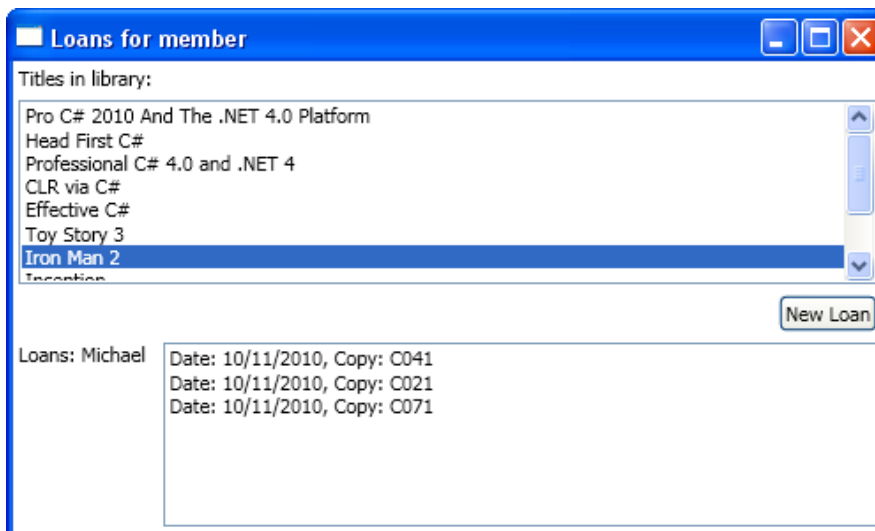


LAB 7: Library system application

Getting started

In this lab you will finally get to create a graphical user interface. You will use the library system model from the previous lab, and build a window using Windows Presentation Foundation (WPF) which allows you to view all the titles in the library catalog and create loans for a specific library member:



Task 1: Window layout with XAML

Open the project:

1. Start a VM which has Visual Studio 2010 installed, and start Visual Studio.
2. Download the file *lab7.zip* from Blackboard, and extract the contents.
3. In Visual Studio, select **File>Open Project..** and browse to find the file *LibraryGUI.sln* inside the *Library* folder, then click **Open**. This solution contains two projects:
 - **Library** – the model classes from the previous lab
 - **LibraryGUI** – a WPF project, which contains one window, defined in *MainWindow.xaml*

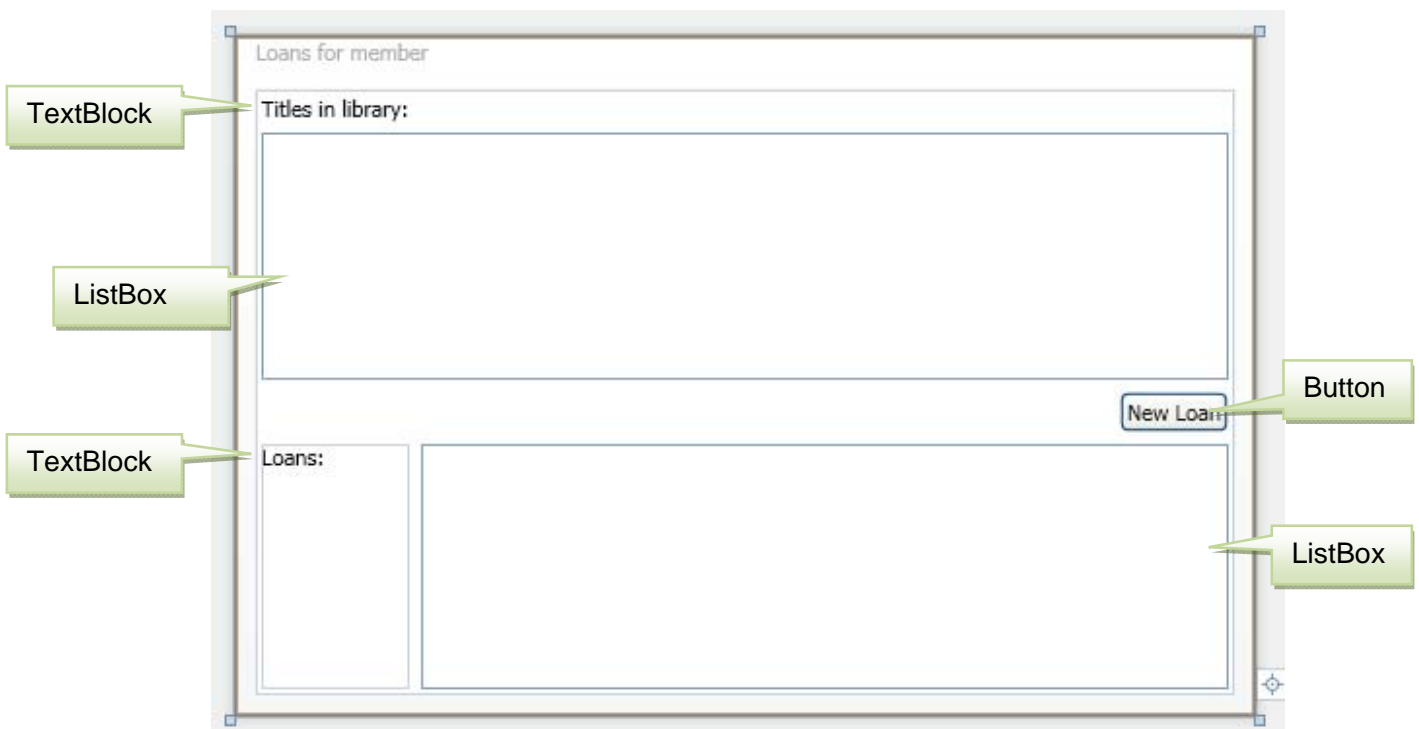
The **Library** project has been added as a reference in the **LibraryGUI** project so that the WPF application can use the model classes. The code-behind file, *MainWindow.xaml.cs* contains the line:

```
using Library;
```

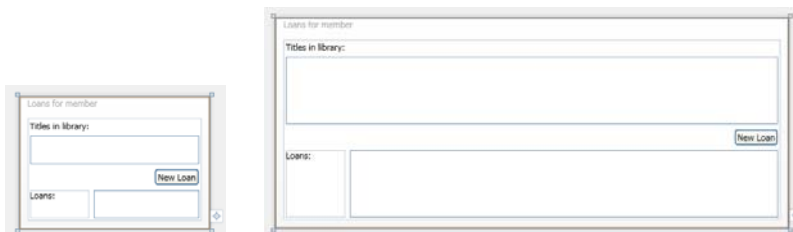
The XAML code in *MainWindow.xaml* contains an empty **<Grid>** element.

Define the layout:

4. **Write XAML code** inside the **<Grid>** element to define a window layout which looks similar to the figure below, with the controls indicated. You should plan out how many rows and columns your grid will have, and how the controls will fit into the grid. Use the Design view to view the layout which your XAML code produces – don't use it to create the layout.



Your layout should be **fluid** – it should cope gracefully with changes to the size of the window. Remember that a user may be able to resize your window when the application is running.



You can build and run the solution to check what it looks like when running.

Task 2: Displaying titles

You will now modify the window so that a list of all the titles in the library is displayed in the upper list box when the application starts.

XAML:

1. In *MainWindow.xaml*, give the upper list box the name *lstTitles* by defining an attribute as follows:

```
<ListBox x:Name="lstTitles"
```

2. The list box should display the *TitleName* property of each of the titles in the library. Define the following attribute for the list box:

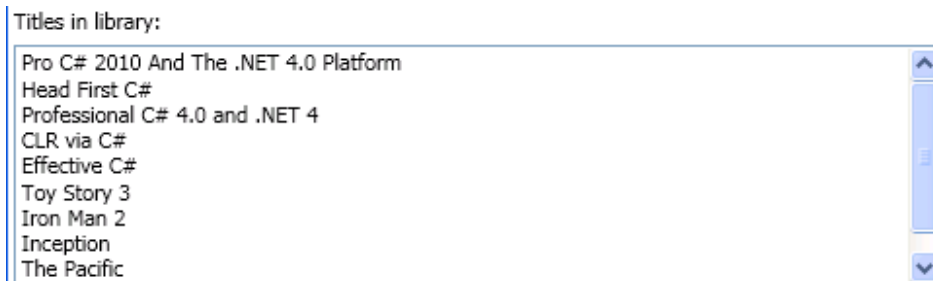
```
DisplayMemberPath="TitleName"
```

Code-behind:

3. Open *MainWindow.xaml.cs* and look at the code. There is a method *GetAllTitles* which creates some *Title* and *Copy* objects, and associates copies with titles. There are ten titles and two copies of each title. The method returns a list of the *Title* objects.
4. Add code in the constructor of *MainWindow* which does the following:
 - Declares a variable *titles* of type *List<Title>* and initialises *titles* by calling *GetAllTitles*.
 - Sets the *ItemsSource* property of the upper list box to *titles*.

Test:

5. Build and run the application. Check that ten title names appear in the list box (there will probably be a scroll bar shown to allow all titles to be viewed).



Task 3: Binding window to a Member object

The purpose of the window is to manage loans for a particular member. You will now bind the window to a specific *Member* object. This *Member* object will become the window's **data context**.

XAML:

1. In *MainWindow.xaml*, add a new *TextBlock* element beside the block which displays "Loans:". You can enclose the two blocks in another container so that they appear neatly side-by-side.
2. The new text block should display the *Name* property of the window's *Member* object. Bind the *Text* property of the text block to the *Name* property of the window data context using the following attribute:


```
Text="{Binding Path=Name, Mode=TwoWay}"
```

Code-behind:

3. In *MainWindow.xaml.cs* there is a method *GetMember* which creates and returns a *Member* object.
4. Add an instance variable called *member*, of type *Member*, to *MainWindow*.
5. Add code in the constructor of *MainWindow* which does the following:
 - Initialises *member* by calling *GetMember*.
 - Sets the *DataContext* property of the window to *member*.

Test:

6. Build and run the application. Check that name of the member appears.

Loans: Michael 

NOTE: The methods *GetTitles* and *GetMember* in this example simply create objects using data coded into the application. More realistically, these methods would retrieve the information needed to create the objects from a database.

Task 4: Creating and displaying loans

In this task you will make the application interactive. When the user clicks the **Create Loans** button the application should create a new *Loan* for the member. The title to be loaned will be the currently selected item in the upper list box. The first available copy of that title will be loaned. The list of loans for the member should be displayed in the lower text box, and this will be updated each time a new loan is created.

XAML:

1. In *MainWindow.xaml*, give the lower list box the name *lstLoans*. Do not define a *DisplayMemberPath* attribute – if no path is specified the return value of the *ToString* method of each displayed object will be shown. You can look at the code for *Loan* to see what *ToString* will return.
2. Name the *Button* control *btnNewLoan* and define an event handler by adding the following attribute to it:

```
Click="btnNewLoan_Click"
```

Code-behind:

3. In *MainWindow.xaml.cs* add an instance variable called *loans*, of type *ObservableCollection<Loan>* to *MainWindow*. *ObservableCollection* is a special collection type that can be bound to a WPF control and will automatically update the control when a new item is added to the collection.
4. Add code in the constructor of *MainWindow* which does the following:
 - Initialises *loans* to an empty *ObservableCollection<Loan>*
 - Sets the *ItemsSource* property of the lower list box to *loans*
5. Add a new event handler method for the button click event using the following code:

```
private void btnNewLoan_Click(object sender,
    RoutedEventArgs e)
{
    Title title = (Title)lstTitles.SelectedItem;
    Copy copy = title.GetCopy();
    Loan newLoan = new Loan(copy, member);
    loans.Add(newLoan);
}
```

Test:

6. Build and run the application. Select “CLR via C#” in the upper list box and click New Loan. Check that a loan for copy C041 is displayed in the lower list box.
7. Repeat for “Toy Story 3”. Check that a loan for copy C061 is displayed in the lower list box.
8. Repeat for “Toy Story 3” again. Check that a loan for copy C062 is displayed in the lower list box.

```
Loans: Michael | Date: 10/11/2010, Copy: C041  
               | Date: 10/11/2010, Copy: C061  
               | Date: 10/11/2010, Copy: C062
```

9. Repeat for “Toy Story 3” again. What happens? Remember that the code in *GetTitles* creates two copies of each title.

Task 5: Exception handling

In lab 6 you saw how the *Title* class was written to throw a custom exception if it can't find an available copy to be loaned. In the last step of Task 4 you should have seen that this exception is not handled in your application. In this task, you will handle the exception in the user interface, and use the information in the custom exception class to give useful information to the user.

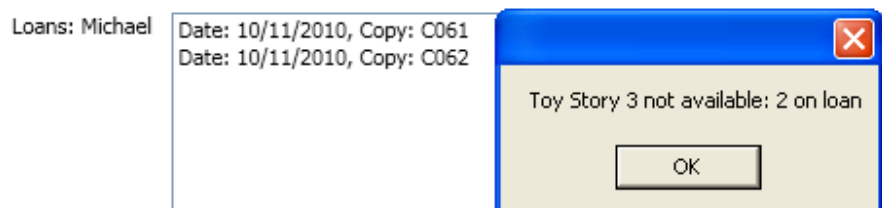
Code-behind:

1. In *MainWindow.xaml.cs* find the button event handler method which you added in task 4. Enclose the code inside the method in a try-catch block, using the following code for the catch part:

```
catch (TitleNotAvailableException ex)
{
    MessageBox.Show (String.Format (
        "{0} not available: {1} on loan",
        ex.TitleName, ex.CopiesOnLoan));
}
```

Test:

2. Build and run the application. Create two loans for "Toy Story 3" as you did in Task 4, and check that the loans are displayed in the lower list box.
3. Click the button to create a third loan for "Toy Story 3". A message box should be shown:



Task 6: Looking at code reuse

You have now completed your WPF application. The GUI code in this application makes use of a **model** which consists of C# classes representing the entities in the library system. These classes implement the **business logic** of the system, while the GUI simply initiates actions and displays results. Because of this, it is easy to re-use the model in a different application which may provide a completely different user interface. In this task you will simply look at a version of the library application which has been created as a web application, using Microsoft's ASP.NET.

Open the project:

1. Download the file *lab7.zip* from Blackboard, and extract the contents.
2. In Visual Studio, select **File>Open Project..** and browse to find the file *LibraryWeb.sln* inside the *Library* folder, then click **Open**. This solution contains two projects:
 - **Library** – the model classes, which are **exactly the same** as in the WPF application – you can look at the code to check this.
 - **LibraryWeb**– an ASP.NET Web project, which contains a home page *Default.aspx*.

The **Library** project has been added as a reference in the **LibraryWeb** project so that the web application can use the model classes.

Test:

1. Build and run (Start Debugging) the application. The home page should be shown in your web browser. Repeat the tests you carried out for the WPF application.

