Пошаговое руководство: Библиотека книг на WPF (.NET 3.5, Visual Studio 2010)

1. Создание проекта

1. Запустите Visual Studio 2010
2. Создайте новый проект: File → New → Project
3. Выберите "WPF Application" (.NET Framework 3.5)
4. Назовите проект "BookLibrary" и нажмите OK

2. Структура проекта

Создайте следующие классы в проекте:

Book.cs (Модель данных)

```csharp
public class Book
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Author { get; set; }
    public string Genre { get; set; }
    public int Year { get; set; }
}
```

RelayCommand.cs (Реализация команд)

```csharp
using System;
using System.Windows.Input;

public class RelayCommand : ICommand
```

```
{
    private readonly Action<object> _execute;
    private readonly Func<object, bool> _canExecute;

    public event EventHandler CanExecuteChanged;

    public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)
    {
        _execute = execute ?? throw new ArgumentNullException(nameof(execute));
        _canExecute = canExecute;
    }

    public bool CanExecute(object parameter)
    {
        return _canExecute == null || _canExecute(parameter);
    }

    public void Execute(object parameter)
    {
        _execute(parameter);
    }

    public void RaiseCanExecuteChanged()
    {
        CanExecuteChanged?.Invoke(this, EventArgs.Empty);
    }
}
```

BookViewModel.cs (ViewModel)

csharp

```
using System;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;

public class BookViewModel : INotifyPropertyChanged
{
    private ObservableCollection<Book> _allBooks;
    private ObservableCollection<Book> _filteredBooks;
    private Book _selectedBook;
    private string _searchText;

    public ObservableCollection<Book> Books
    {
        get => _filteredBooks;
        private set
        {
            _filteredBooks = value;
            OnPropertyChanged(nameof(Books));
        }
    }

    public Book SelectedBook
    {
        get => _selectedBook;
        set
        {
            _selectedBook = value;
            OnPropertyChanged(nameof(SelectedBook));
            DeleteCommand.RaiseCanExecuteChanged();
            EditCommand.RaiseCanExecuteChanged();
```

```
        }
    }

    public string SearchText
    {
        get => _searchText;
        set
        {
            _searchText = value;
            OnPropertyChanged(nameof(SearchText));
            FilterBooks();
        }
    }

    public RelayCommand AddCommand { get; }
    public RelayCommand DeleteCommand { get; }
    public RelayCommand EditCommand { get; }

    public BookViewModel()
    {
        _allBooks = new ObservableCollection<Book>
        {
            new Book { Id = 1, Title = "Гарри Поттер и философский камень", Author = "Джоан Роулинг", Genre = "Фэнтези", Year = 1997 },
            new Book { Id = 2, Title = "Властелин колец", Author = "Дж. Р. Р. Толкин", Genre = "Фэнтези", Year = 1954 },
            new Book { Id = 3, Title = "Убить пересмешника", Author = "Харпер Ли", Genre = "Роман", Year = 1960 },
            new Book { Id = 4, Title = "Великий Гэтсби", Author = "Фрэнсис Скотт Фицджеральд", Genre = "Классика", Year = 1925 },
            new Book { Id = 5, Title = "451 градус по Фаренгейту", Author = "Рэй Брэдбери", Genre = "Антиутопия", Year = 1953 }
        };
```

```
    _filteredBooks = new ObservableCollection<Book>(_allBooks);

    AddCommand = new RelayCommand(_ => AddBook());
    DeleteCommand = new RelayCommand(_ => DeleteBook(), _ => SelectedBook != null);
    EditCommand = new RelayCommand(_ => EditBook(), _ => SelectedBook != null);
}

private void AddBook()
{
    var newBook = new Book
    {
        Id = _allBooks.Count > 0 ? _allBooks.Max(b => b.Id) + 1 : 1,
        Title = "Новая книга",
        Author = "Новый автор",
        Genre = "Новый жанр",
        Year = DateTime.Now.Year
    };

    _allBooks.Add(newBook);
    FilterBooks();
    SelectedBook = newBook;
}

private void DeleteBook()
{
    if (SelectedBook != null)
    {
        _allBooks.Remove(SelectedBook);
        FilterBooks();
    }
}
```

```
private void EditBook()
{
    if (SelectedBook != null)
    {
        // В реальном приложении здесь можно открыть диалог редактирования
        var editedBook = SelectedBook;
        editedBook.Title += " (изд.)";
        FilterBooks();
    }
}

private void FilterBooks()
{
    if (string.IsNullOrWhiteSpace(SearchText))
    {
        Books = new ObservableCollection<Book>(_allBooks);
    }
    else
    {
        var searchLower = SearchText.ToLower();
        var filtered = _allBooks.Where(b =>
            b.Title.ToLower().Contains(searchLower) ||
            b.Author.ToLower().Contains(searchLower) ||
            b.Genre.ToLower().Contains(searchLower));

        Books = new ObservableCollection<Book>(filtered);
    }
}

public event PropertyChangedEventHandler PropertyChanged;
```

```
    protected virtual void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

3. Реализация интерфейса (MainWindow.xaml)

```
xml
<Window x:Class="BookLibrary.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Библиотека книг" Height="450" Width="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <!-- Панель поиска -->
        <StackPanel Grid.Row="0" Orientation="Horizontal" Margin="5">
            <TextBlock Text="Поиск:" VerticalAlignment="Center" Margin="0,0,5,0"/>
            <TextBox Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"
                     Width="200"/>
        </StackPanel>

        <!-- Список книг -->
        <ListView Grid.Row="1" ItemsSource="{Binding Books}" SelectedItem="{Binding SelectedBook}"
                  Margin="5">
```

```
        <ListView.View>
            <GridView>
                <GridViewColumn Header="ID" DisplayMemberBinding="{Binding Id}" Width="50"/>
                <GridViewColumn Header="Название" DisplayMemberBinding="{Binding Title}"
Width="200"/>
                <GridViewColumn Header="Автор" DisplayMemberBinding="{Binding Author}"
Width="150"/>
                <GridViewColumn Header="Жанр" DisplayMemberBinding="{Binding Genre}" Width="100"/>
                <GridViewColumn Header="Год" DisplayMemberBinding="{Binding Year}" Width="60"/>
            </GridView>
        </ListView.View>
    </ListView>

    <!-- Панель кнопок -->
    <StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Right" Margin="5">
        <Button Content="Добавить" Command="{Binding AddCommand}" Width="80"
Margin="0,0,5,0"/>
        <Button Content="Удалить" Command="{Binding DeleteCommand}" Width="80"
Margin="0,0,5,0"/>
        <Button Content="Редактировать" Command="{Binding EditCommand}" Width="100"/>
    </StackPanel>
  </Grid>
</Window>
```

4. Настройка DataContext (MainWindow.xaml.cs)

```
csharp
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
```

```
        DataContext = new BookViewModel();
    }
}
```

5. Создание тестов (MS Test)

Добавьте новый проект в решение:

1. File → Add → New Project
2. Выберите "Test Project" (.NET Framework 3.5)
3. Назовите "BookLibrary.Tests"

BookViewModelTests.cs

```
csharp
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Linq;

[TestClass]
public class BookViewModelTests
{
    [TestMethod]
    public void AddBook_ShouldIncreaseCount()
    {
        var vm = new BookViewModel();
        int initialCount = vm.Books.Count;
        vm.AddCommand.Execute(null);
        Assert.AreEqual(initialCount + 1, vm.Books.Count);
    }

    [TestMethod]
```

```
    public void DeleteBook_WithSelection_ShouldDecreaseCount()
    {
        var vm = new BookViewModel();
        vm.SelectedBook = vm.Books.First();
        int initialCount = vm.Books.Count;
        vm.DeleteCommand.Execute(null);
        Assert.AreEqual(initialCount - 1, vm.Books.Count);
    }

    [TestMethod]
    public void DeleteBook_WithoutSelection_ShouldNotExecute()
    {
        var vm = new BookViewModel();
        vm.SelectedBook = null;
        Assert.IsFalse(vm.DeleteCommand.CanExecute(null));
    }

    [TestMethod]
    public void Search_FilterBooks_Correctly()
    {
        var vm = new BookViewModel();
        vm.SearchText = "Толстой";
        Assert.IsTrue(vm.Books.All(b => b.Author.Contains("Толстой")));
    }
}
```

6. Запуск и тестирование

1. Нажмите F5 для запуска приложения
2. Проверьте функциональность:

- Добавление новых книг

- Удаление книг

- Поиск по названию, автору или жанру

3. Запустите тесты через меню Test → Run → All Tests

7. Дополнительные улучшения

Для полноценного приложения можно добавить:

1. Диалог редактирования книги

2. Сохранение данных в файл (XML или JSON)

3. Валидацию ввода

4. Сортировку по колонкам