LINQ to XML. Учебные задания. Часть 2

Описание интерфейса

Каждый вариант индивидуальных заданий предполагает разработку оконного графического WPF-приложения, позволяющего перемещаться по каталогам текущего диска, выбирать в них файлы формата fb2 и обрабатывать их в соответствии с условиями заданий, входящих в данный вариант.

Окно графического приложения должно иметь вид, приведенный на рис. 1 (первые три вкладки правой панели должны использоваться в любом варианте индивидуальных заданий).

Рис. 1. Окно программы с выбранной вкладкой **Title-info**

Основные этапы разработки интерфейса, обеспечивающего навигацию по каталогам текущего диска, описаны в проекте IMGVIEW книги «Разработка пользовательского интерфейса на основе технологии Windows Presentation Foundation» (гл. 17).

Для создания набора вкладок следует использовать компонент `TabControl`. Каждая вкладка оформляется как дочерний элемент-объект `TabItem` компонента `TabControl`. Для каждого элемента `TabItem` надо задать атрибут `Header` и дочерний элемент, определяющий содержимое данной вкладки.

Поскольку в программном коде требуется многократно обращаться к текущему документу, а также использовать пространство имен, определенное в корневом элементе и связанное со всеми элементами документа, рекомендуется описать в классе `MainWindow` два поля:

```
XDocument d = null;
  // текущий загруженный XML-документ
XNamespace ns = null;
  // пространство имен элементов XML-документа
```

Определение этих полей надо выполнять в обработчике `fileList1_SelectionChanged`:

```
var name = ((dirList1.SelectedItem as TreeViewItem)
   .Tag as DirectoryInfo).FullName + "\\" +
   (string)fileList1.SelectedValue;
d = XDocument.Load(name);
ns = d.Root.Name.Namespace;
```

В этом же обработчике надо вызывать все методы, обеспечивающие обновление содержимого вкладок на правой панели.

При смене каталога требуется автоматически выбирать первую вкладку. При наличии в каталоге книг на вкладке сразу отображается информация о первой книге; если каталог не содержит книг формата fb2, то вкладка должна быть пустой и, кроме того, набор вкладок должен стать недоступным.

Все многострочные поля ввода, используемые в проекте (в частности, поле `textBox1`, размещенное на вкладке **Title-info**), должны быть доступны только для чтения. Кроме того, они должны обеспечивать разбиение длинных строк на части и при необходимости содержать вертикальную полосу прокрутки.

Программа должна сохранять информацию о своих текущих настройках (размеры окна, ширина панелей, текущий каталог и текущий файл) в реестре Windows и использовать ее при последующем запуске (см. два последних пункта главы 17, посвященной проекту IMGVIEW).

Вспомогательный запрос LINQ для объединения последовательностей

При выполнении некоторых заданий требуется объединять строковые представления элементов последовательности, добавляя между ними символ-разделитель (например пробел). Поскольку подобный вариант запроса агрегирования не входит в стандартный набор запросов LINQ to Objects, в программе целесообразно описать новый запрос, реализованный в виде метода расширения:

```
public static class ExtCombine
{
  public static string Combine<T>(this
    IEnumerable<T> src, string separator)
  {
    if (!src.Any())
      return null;
    return src.Aggregate("", (seed, s) => seed +
      s.ToString() + separator,
      s => separator.Length > 0 ?
      s.Remove(s.Length - separator.Length) : s);
  }
}
```

Класс `ExtCombine` можно описывать как до, так и после класса `MainWindow`.

Задание 1. Вывод сводной информации

Формулировка

Отобразить на вкладке **Title-info** сводную информацию о текущей книге, используя содержимое элемента `title-info` соответствующего XML-документа. Образец сводной информации приведен на рис. 1.

Указания к выполнению

В сводную информацию включаются все дочерние элементы элемента `title-info`. Информация о каждом элементе выводится с новой строки и начинается с локального имени элемента, после которого следует двоеточие (имя пространства имен не выводится). Способ отображения содержимого элемента зависит от его структуры. Возможны три варианта способа отображения:

- если элемент имеет дочерние элементы (например `author`); то отображаются значения всех его дочерних элементов, разделенные пробелами; в случае, если результат является пустой строкой (например, для элемента `coverpage`), информация о данном элементе не выводится;
- если элемент не имеет дочерних элементов, но имеет непустое содержимое (например `book-title`), то выводится это содержимое;
- если элемент не имеет дочерних элементов и его содержимое является пустым, однако он имеет непустой набор атрибутов (примером такого элемента является `sequence`), то его атрибуты сортируются в алфавитном порядке их имен и выводятся на новой строке в формате `имя="значение"` (между атрибутами указывается пробел).

Дочерние элементы должны обрабатываться в том порядке, в котором они расположены в содержащем их родительском элементе.

Задание 2. Редактирование сводной информации

Формулировка

С помощью текстовых полей, размещенных на вкладке **Edit**, обеспечить возможность редактирования некоторых элементов из сводной информации о текущей книге и сохранение измененных значений элементов в исходном XML-файле.

Данное задание может быть предложено в трех вариантах.

2-1. Обеспечить возможность редактирования элементов `author` (если имеется несколько авторов, то редактируется первый из них) и `book-title`.

2-2. Обеспечить возможность редактирования элементов `book-title` и `sequence`.

2-3. Обеспечить возможность редактирования элементов `author` (если имеется несколько авторов, то редактируется первый из них), `book-title` и `sequence`.

Вид вкладки **Edit** для задания 2-3 приведен на рис. 2.

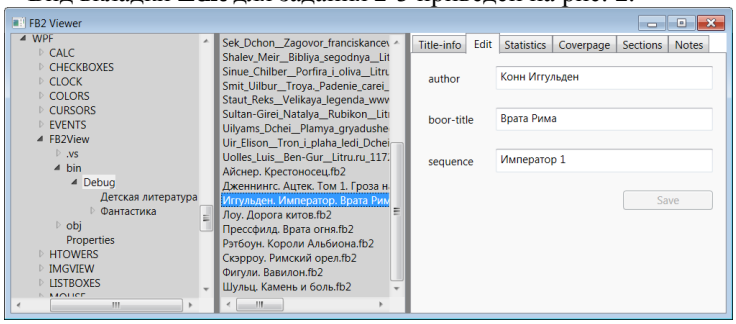

Рис. 2. Окно программы с выбранной вкладкой **Edit**

Указания к выполнению

В качестве группирующего компонента для вкладки **Edit** используйте компонент `Grid` с 4 строками и 2 столбцами.

Для редактирования предназначены только элементы, являющиеся дочерними элементами элемента `title-info` (следует заметить, что одноименные элементы могут входить и в другие элементы XML-документа, например, в элемент `document-info`).

При любом изменении полей ввода кнопка **Save** делается активной. При попытке перейти к другому документу, сменить текущий каталог или выйти из программы при активной кнопке **Save** на экран должен выводиться запрос о сохранении измененных данных (для этого используется метод `MessageBox.Show`). После сохранения данных кнопка **Save** опять делается неактивной.

Перед обработкой измененных данных в них необходимо удалить начальные, конечные и повторяющиеся пробелы. Если в результате данные об авторе или о названии книги станут пустыми, то соответствующие элементы XML-документа изменять не следует. Указание пустой строки в поле ввода `sequence` (данные о *серии* книги) должно приводить к удалению соответствующего элемента из XML-документа.

После сохранения измененных данных должно немедленно обновляться содержимое как вкладки **Title-info**, так и вкладки **Edit** (в частности, если поле `author` или `book-title` было сделано пустым, то в нем опять должно отображаться его прежнее содержимое).

При изменении информации об авторе должны учитываться следующие правила:

- если информация состоит из одного слова (*словом* считается последовательность символов, не содержащая пробелов и ограниченная пробелами, началом или концом строки), то это слово считается фамилией (элемент `last-name`);
- если информация состоит из двух слов, то первое слово считается именем (`first-name`), а второе слово — фамилией (`last-name`);
- если информация состоит более чем из двух слов, то все слова, начиная со второго и заканчивая предпоследним, вместе с разделяющими их пробелами, относятся к элементу `middle-name`, первое слово считается именем, а последнее — фамилией.

При обновлении информации об авторе следует учитывать, что в результирующем XML-документе должен сохраняться стандартный порядок следования дочерних элементов: `first-name`, `middle-name`, `last-name`. Элементы с пустым содержимым в документ включать не следует.

При изменении информации о серии должны учитываться следующие правила:

- если информация состоит из одного слова, то оно считается названием серии (атрибут `name`), а номер (элемент `number`) в этом случае полагается равным 0;
- если информация состоит более чем из одного слова, то все слова, начиная с первого и заканчивая предпоследним, вместе с разделяющими их пробелами, относятся к атрибуту `name`, а последнее слово — к атрибуту `number`.

Проверять, что атрибут `number` можно преобразовать в число, не требуется. Порядок записи атрибутов в элемент `sequence` может быть произвольным.

Задание 3. Вывод статистики, связанной с элементами и атрибутами XML-документа

Формулировка

Отобразить на вкладке **Statistics** информацию, связанную с использованием различных элементов и/или атрибутов в текущем XML-документе.

Данное задание может быть предложено в трех вариантах.

3-1. Вывести статистику использования элементов различных видов в документе XML.

3-2. Вывести статистику использования атрибутов различных видов в документе XML.

3-3. Вывести статистику использования как элементов, так и атрибутов различных видов в документе XML.

Указания к выполнению

Информация, связанная с элементами или атрибутами XML, должна начинаться со строки «Elements:» или «Attributes:» соответственно.

В список элементов (атрибутов) надо включить локальные имена всех элементов (соответственно, атрибутов), указав после каждого имени количество вхождений в документ элемента (атрибута) с данным именем. Количество вхождений отделяется от имени текстом « - » (дефис и обрамляющие его пробелы).

Информация о каждом элементе (атрибуте) выводится на отдельной строке; элементы (атрибуты) выводятся в алфавитном порядке их имен.

В статистику следует включать и данные, полученные для корневого элемента `FictionBook` и его атрибутов.

Задание 4. Отображение специальных элементов книги

Формулировка

Отобразить на дополнительных вкладках информацию о специальных элементах книги.

Данное задание может быть предложено в трех вариантах.

4-1. Отобразить на вкладке **Coverpage** изображение обложки книги, если соответствующий элемент (`coverpage`) и связанные с ним элементы имеются в текущем XML-документе.

4-2. Отобразить на вкладке **Sections** оглавление книги, используя вложенные элементы `section`, входящие в *первый* элемент `body`.

4-3. Отобразить на вкладке **Notes** примечания книги, используя дочерние элементы элемента `body`, имеющего атрибут `name` со значением `notes`.

Виды вкладок **Coverpage**, **Sections** и **Notes** приведены на рис. 3–5.

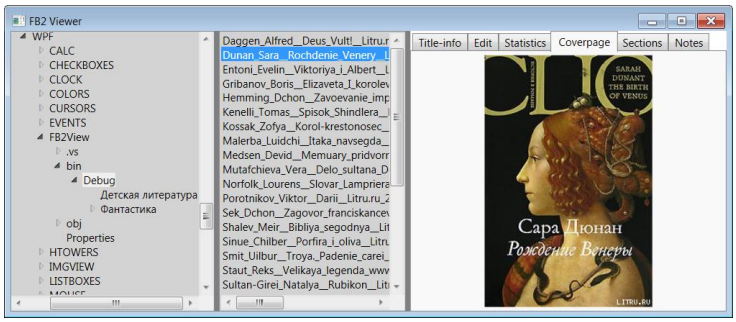

Рис. 3. Окно программы с выбранной вкладкой **Coverpage**

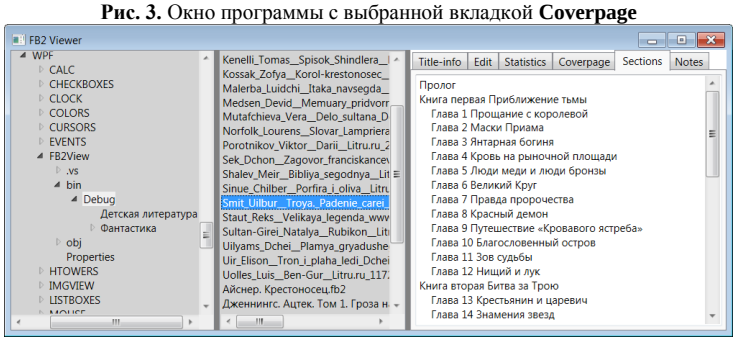

Рис. 4. Окно программы с выбранной вкладкой **Sections**

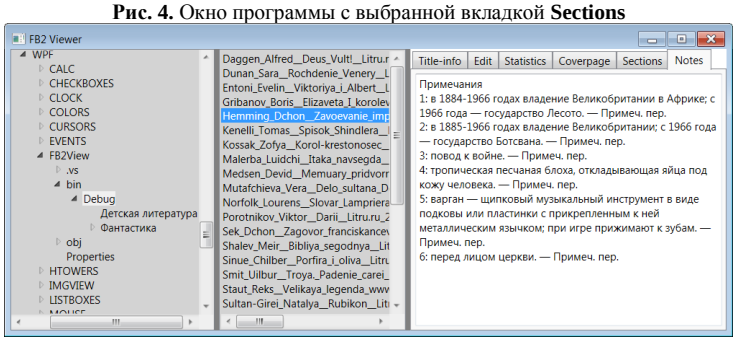

Рис. 5. Окно программы с выбранной вкладкой **Notes**

Указания к выполнению

Для вывода изображения следует использовать компонент Image, установив для него свойство Stretch равным Uniform (что обеспечит пропорциональное масштабирование изображения в пределах вкладки **Coverpage**).

Для того чтобы вывести на вкладке изображение обложки, проще всего предварительно сохранить это изображение во вспомогательном файле, после чего загрузить его в компонент Image с помощью объекта `BitmapImage` (см. проект IMGVIEW, п. 17.3). Для возможности перезаписи вспомогательного файла следует связывать объект `BitmapImage` с копией содержимого этого файла, загруженной в память (см. комментарий 1 к указанному пункту).

Пример кода, обеспечивающего сохранение во внешнем файле двоичных данных из XML-документа, был приведен в пункте «Хранение в XML-документе двоичных данных». Следует также учитывать, что для поиска элемента `binary`, соответствующего обложке книги, необходимо использовать атрибут `href`, снабженный префиксом специального пространства имен `http://www.w3.org/1999/xlink` (см. последний пример, приведенный в пункте «Префиксы пространства имен»).

Если текущий XML-документ не содержит элемента `coverpage`, то вкладка **Coverpage** должна быть пустой.

При смене текущего документа (и при завершении программы) вспомогательный файл с изображением обложки должен быть удален (методом `File.Delete`).

При формировании оглавления следует учитывать, что оглавление может иметь несколько уровней (это обеспечивается использованием *вложенных* элементов `section`). Выделение уровней надо выполнять с помощью отступов; отступ для следующего уровня должен быть на 4 пробела больше отступа предыдущего уровня. Для первого уровня отступы не используются. Учитывая произвольную глубину вложенности уровней оглавления, для его формирования целесообразно использовать вспомогательный рекурсивный метод.

Название текущего раздела, оформленного в виде элемента `section`, содержится в его дочернем элементе `title`, который, в свою очередь, содержит один или несколько элементов `p`. Название следует формировать из значений всех элементов `p`, являющихся дочерними элементами элемента `title`, разделяя их пробелами. Если некоторые из элементов `section` не содержат дочернего элемента `title`, то они не должны включаться в оглавление.

Элемент `body` с атрибутом `name="notes"`, используемый для хранения примечаний к книге, может содержать элемент `title` — заголовок списка примечаний (как правило, со значением «Примечания»). Сами примечания оформляются как подразделы списка примечаний, с использованием элементов `section`. Каждый из элементов `section`, в свою очередь, состоит из элемента `title` — заголовка примечания (как правило, с числовым значением) и одного или нескольких элементов `p`, содержащих текст примечания.

При формировании списка примечаний следует вывести на отдельной строке заголовок списка примечаний (или пустую строку, если элемент `title` в элементе `body` отсутствует). Каждое примечание также выводится на отдельной строке; вначале указывается его заголовок, затем двоеточие и пробел, после чего выводится текст примечания. Если текст примечания состоит из нескольких абзацев (т. е. элементов `p`), то каждый новый абзац выводится на новой строке.

Если текущий XML-документ не содержит примечаний, то вкладка **Notes** должна быть пустой.

Варианты индивидуальных заданий

Вариант 1: 1 2-1 3-1 4-1
Вариант 2: 1 2-1 3-1 4-2
Вариант 3: 1 2-1 3-1 4-3
Вариант 4: 1 2-1 3-2 4-1
Вариант 5: 1 2-1 3-2 4-2
Вариант 6: 1 2-1 3-2 4-3
Вариант 7: 1 2-2 3-1 4-1
Вариант 8: 1 2-2 3-1 4-2
Вариант 9: 1 2-2 3-1 4-3
Вариант 10: 1 2-2 3-2 4-1
Вариант 11: 1 2-2 3-2 4-2
Вариант 12: 1 2-2 3-2 4-3