Оглавление

Глава 1

XAML как XML-приложение

Язык декларативного программирования XAML (eXtensible Application Markup Language – произносится как «зэмл», а переводится как «расширяемый язык разметки приложений») является одним из языков, построенных на основе языка XML (eXtensible Markup Language – «расширяемый язык разметки»).

Глубокого изучения XML для понимания синтаксиса XAML не требуется, но нужно знать, что такое XML-документ, как определяются его элементы и атрибуты. Начнем с того, что XML (eXtensible Markup Language) – это не язык разметки, как, например, HTML, а основа и средство для разработки специализированных языков разметки.

Разметка (markup) – это добавление в текст документа специальных дескрипторов (именованных меток) для обозначения и выделения частей документа. Упомянутые дескрипторы разметки называются тегами. В общем случае выделяемый или размечаемый с помощью XML-средств фрагмент (участок) текста ограничен двумя тегами:

<*имя_тега*> и </*имя_тега*>.

Первый из них называют начальным тегом, в нем, кроме имени тега, могут помещаться атрибуты, разделяемые пробелами. Второй из тегов называют конечным тегом.

С помощью тегов:

- указываются границы участка (фрагмента) текста;
- определяется роль (назначение) выделяемого фрагмента в тексте документа;
- определяется расположение (позиция) фрагмента относительно других участков текста;
- определяется вложенность фрагментов текста;

- определяются связи фрагмента текста с ресурсами, размещёнными за пределами документа.

Фрагмент между начальным и конечным тегами вместе с этими тегами называют XML-элементом. При XML-разметке весь размечаемый текст ограничивается начальным и конечным тегами, которые определяют размечаемый текст как отдельный XML-документ. XML-документ называют корневым элементом (root element). В корневой элемент XML-документа обычно включают некоторый текст документа и набор элементов более низкого уровня. Каждый элемент вводится и ограничивается своей парой тегов. Имя тега определяет имя или название соответствующего XML-элемента. Таким образом, XML-документ в общем случае – это дерево элементов, в котором XML-документ – это основная единица или корень дерева.

В начальный тег каждого элемента (как уже упомянуто), кроме его названия, обычно включают атрибуты, определяющие какие-либо свойства элемента. Каждый атрибут имеет уникальное для данного тега имя и значение, отделяемое от имени атрибута знаком =. Значение атрибута задается в виде строки, ограниченной кавычками или апострофами. Количество атрибутов может быть произвольным. Их имена зависят от вида (типа) определяемого элемента.

Приведем синтаксически верный XML-документ:

```
<!-- №_01_00.xml -->
<Label FontSize = "30" Background="LightGray">
   Label - это "ярлык"!
</Label>
```

Первая строка документа – комментарий XML. Он начинается лексемой из четырех символов (<!--) и завершается лексемой из трех символов (-->). Между ними – достаточно произвольный текст. В приведенном XML-документе – только один элемент, вводимый тегом с именем Label. Зная, как используется элемент управления Label в технологиях WinForms и WPF, название этого элемента можно, наверное, переводить на русский язык как «ярлык» или «наклейка с надписью». Часто этот элемент называют «меткой», но для декларативных языков (к числу которых относится XML и созданные на его основе языки разметки) этот перевод мало подходит по смыслу. Вводя ярлык, нужно указать, что будет на нем написано (содержимое), каким шрифтом, на каком фоне и т. п. Именно

эта информация в данном случае определена в приведенном документе. Между его начальным тегом <Label…> и конечным тегом </Label> помещено содержимое элемента – в данном случае текст «Label – это"ярлык"!». В начальном теге для элемента Label определены два разделенных пробелом атрибута: FontSize и Background. Первый определяет размер шрифта текста, используемый при визуализации документа, второй задает цвет фона изображения элемента Label. Обратите внимание, что значения атрибутов заключены в кавычки. (Возможно применение и апострофов.)

Хотя приведенный текст соответствует правилам оформления XML-документа, но программа его обработки должна (по крайней мере) «знать», что означают использованные в декларации документа идентификаторы (Label, FontSize, Background и т. п.). Как же это становится известно программе, и какие программы могут правильно интерпретировать XML-документ?

В отличие от предшествующих языков, XML не является сам по себе языком разметки. Он не определяет теги разметки, а даёт возможность определять эти теги пользователю XML. Пользователь может вводить собственные имена тегов, либо брать за основу кем-то созданный язык разметки и применять, либо расширять набор тегов этого чужого языка.

Однако, создавая собственный язык разметки, его автор должен следовать строгим правилам языка XML. Одно из них заключается в требовании к структуре документа. Структура документа должна однозначно определять правила его интерпретации разными XML-процессорами. Именно документ с такой структурой считается корректным с точки зрения XML-языка (well_formed).

Язык разметки, созданный с помощью XML, называют XML-приложением (XML application). В настоящее время существует очень много XML-приложений, каждое из которых создано для тех или иных целей.

Документ, размеченный с помощью любого из XML-приложений, должен соответствовать минимальному набору правил XML. Для соблюдения этих правил и правил конкретного языка разметки предусматривается создание спецификации создаваемого языка разметки. Спецификация представляет собой набор требований, которым должен соответствовать правильно размеченный документ.

При разметке документа требуется определить его *структуру* и его *представление*. Представление (presentation) определяет внешний вид документа при его визуализации. В таком языке, как

HTML (который появился раньше, чем был разработан XML), правила представления документа и правила идентификации его элементов объединены. Точнее, набор тегов HTML содержит как теги объявления элементов, так и теги, определяющие их представление при отображении. К первым, т. е. к тегам разметки, в HTML относятся, например, теги, выделяющие заголовок документа или его тело. Представление, то есть формат отображения документа или его фрагмента, в HTML определяют теги, указывающие, например, нужный шрифт вывода текста.

В XML объединение тегов разметки с тегами представления элементов, по крайней мере, *не одобряется*. Не приводя примеров затруднений, которые возникают при одновременном включении в язык разметки как тегов, определяющих структуру документа, так и тегов для задания формата представления, отметим, что корректные XML-документы не должны содержать описаний форматов отображения элементов.

XML-документ определяет с помощью тегов разметки структуру документа, а форматы отображения элементов размещаются в отдельном документе (или в отдельном фрагменте документа), названном таблицей стилей (хотя термин «Style sheet» было бы точнее перевести как «лист стилей», но в русскоязычной литературе установилось название «таблица стилей»). Такое выделение таблицы стилей позволяет, меняя таблицу стилей, по-разному представлять документ, например, при выводе на печать и при отображении на экране.

Пространства имен

Так как в крупных проектах могут быть совместно использованы фрагменты, закодированные с помощью разных языков разметки, то возможно совпадение имен тегов из разных языков разметки. Для исключения такой возможной неоднозначности в XML введен механизм пространств имен. Конкретное пространство имен объявляется в XML-документе при помощи атрибута xmlns. Атрибут xmlns помещают в начальном теге того элемента, в котором использована разметка на соответствующем языке. Объявление пространства имен относится к тому элементу, в котором использован этот атрибут, и ко всем вложенным в него элементам. Название пространства имен должно быть уникальным для каждого XML-приложения.

Не углубляясь более в общие особенности построения XML-приложений, укажем только, что язык XAML – только одно из XML-приложений, причем существуют подмножества этого языка, ориентированные на применения в разных технологиях. (Назовем: Silverlight; WCF – Windows Communication Foundation; WPF – Windows Presentation Foundation; WWF – Windows Workflow Foundation; WCS – Windows CardSpace. И наконец, наиболее свежие: Universal Windows Platform – UWP и Xamarin.Forms.)

Для обозначения пространства имен при применении XAML в среде выполнения ОС Windows используется атрибут:

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

Строка – значение атрибута xmlns – это просто название, которое фирма Microsoft ввела для обозначения того пространства имен, которому «приписаны» элементы языка XAML в среде выполнения ОС Windows. Пространство имен xmlns определяет термины словаря языка XAML (Label, Button, …), относящиеся к среде исполнения ОС Windows. При наличии в документе такого объявления пространства имен xmlns-термины (имена элементов и атрибутов) могут использоваться в декларации XAML-документа по умолчанию (то есть без явного указания того, что термины принадлежат именно к этому пространству имен). Обратим внимание, что строка, определяющая пространство имен xmlns, не имеет никакого отношения к интернет-адресации, несмотря на наличие префикса «http:». Добавление этого атрибута xmlns в начальный тег приведенного выше синтаксически верного XML-документа сделает его XAML-документом, пригодным для обработки и выполнения в среде ОС Windows:

```
<!-- №_01_01.xaml - автономный XAML-документ -->
<Label FontSize = "30" Background="LightGray"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
   Label - это "ярлык"!
</Label>
```

Автономные XAML-документы

Если сохранить приведенный документ в текстовом файле с расширением «.xaml» (используя кодировку UTF-8), то получим автономный XAML-файл или «автономный XAML-документ». Автономный XAML-файл можно обработать и визуализировать,

например, веб-браузером Internet Explorer (см. рис. 1.1). Для этого достаточно указать в браузере путь к файлу или щелкнуть мышью на имени файла в каталоге.

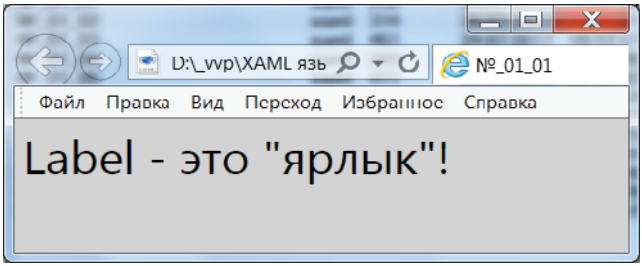

Рис. 1.1. Отображение автономного XAML-файла браузером

Следует отметить, что в ряде случаев для корректного отображения XAML-документа потребуется «настройка» браузера Internet Explorer.

Последовательность шагов настройки браузера:

1. В строке меню браузера выберите «Сервис».
2. В ниспадающем меню нажмите «Свойства браузера».
3. В диалоговом окне свойств браузера выберите «Безопасность».
4. В поле безопасности щелкните кнопку «Другой...».
5. В окне параметров безопасности установите флажки «Включить» для пункта «XAML-приложения веб-обозревателя» и для пункта «Свободный XAML».
6. Щелкните на кнопке «ОК».
7. Во всплывающем окне разрешите изменить настройку.
8. Для завершения настроек щелкните на кнопке «ОК».

Обратите внимание (см. рис. 1.1), что текст содержимого элемента Lable размещен в левом верхнем углу окна браузера, и все окно окрашено в цвет, определенный атрибутом Background. Немного усовершенствуем XAML-код. Добавим к элементу Label еще два атрибута: VerticalAlignment="Center" и HorizontalAlignment="Center". Эти атрибуты «заставят» размещать изображение элемента Label (вместе с содержимым) в центре окна браузера. Автономный XAML-документ примет вид:

```
<!-- №_01_02.xaml – автономный XAML-документ -->
<Label
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
   FontSize = "30"
   VerticalAlignment = "Center"
   HorizontalAlignment = "Center"
   Background="LightGray">
      Label - это "ярлык"!
</Label>
```

Сохраним полученный XAML-документ в файле №_01_02.xaml (используя кодировку UTF-8). Его представление в окне браузера показано на рис. 1.2.

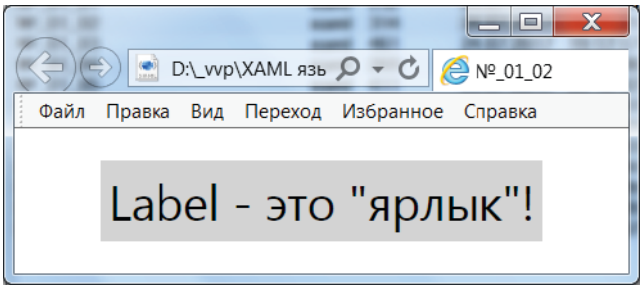

Рис. 1.2. Представление автономного XAML-файла в окне браузера

Обратите внимание, что в этом случае для размещения текста из ярлыка (из элемента Lable) выделен прямоугольник, и только он закрашен цветом, определенным атрибутом Background. Каким образом можно явно определить размеры этого прямоугольника, рассмотрим позже. В данном документе они выбираются по умолчанию с учетом размеров надписи.

Синтаксис XML и синтаксис XAML

Прежде чем переходить к XAML, еще раз остановимся на тех особенностях *синтаксиса XML*, которые нужно учитывать, используя XAML-разметку.

Итак, нам понятен формат декларации (объявления) XML-элемента:

*<**имя_элемента** атрибуты>*
Содержимое
*</**имя_элемента**>*

Началом объявления элемента служит начальный тег с именем элемента. В конце декларации элемента – закрывающий тег с именем того же элемента. Особое назначение в тексте разметки

имеют символы: пробел, табуляция, перевод строки, знак равенства (присваивания), кавычка, апостроф и символы < («меньше») и > («больше»). Эти символы зарезервированы и играют служебные роли. Они используются в качестве разделителей имен элементов, имен атрибутов и значений атрибутов, поэтому эти символы недопустимы в именах элементов. Закрывающий тег никогда не имеет атрибутов, и его имя совпадает с именем элемента. Если в элементе отсутствует содержимое (или оно вводится с помощью специального атрибута, как будет показано позже в примерах XAML-документов), то такой «пустой» элемент можно описать одним тегом:

<*имя_элемента атрибуты*/>

Это так называемая самозакрывающаяся форма записи тега.

Между открывающей угловой скобкой тега (<) и именем элемента не должно быть пробельных символов, но лишние пробелы в любом другом месте тега и в декларации элемента допустимы. Это позволяет разбивать объявление элемента на несколько строк, что продемонстрировано выше в тексте примера №_01.xaml.

При разметке документа нужно соблюдать правила вложения его элементов. Закрывающий тег элемента должен помещаться после открывающего тега. Открывающий и закрывающий теги элемента должны размещаться в пределах охватывающего их элемента более «высокого» уровня.

Внутри объявления элемента (между тегами) все, что не соответствует по синтаксису объявлениям вложенных элементов, воспринимается как текст. В тексте не должны присутствовать символы, которые «зарезервированы» для специальных целей. Например, символ < воспринимается как начало тега и не может применяться непосредственно в тексте. Таким образом, поместив в качестве содержимого в документ текст: «3 < 5», получим ошибку при воспроизведении документа.

В случае необходимости для правильного представления в содержимом зарезервированных символов используются подстановки вида «&*код*;». Такие конструкции иногда называют эскейп-последовательностями. В руководствах по XML такую подстановку называют символьная сущность (character entity). Код в эскейп-последовательности – это условное обозначение (мнемоника) зарезервированного символа либо символ #, за которым следует числовое значение (десятичное или шестнадцатеричное целое

число). Чтобы не отсылать читателя к таблицам специальных символов (см., например, https://dev.w3.org/html5/html-author/charref), приведем наиболее полезные для разметки XML- и XAML-документов:

- & или & – амперсанд (&);
- > или > – правая угловая скобка (>);
- < или < – левая угловая скобка (<);
- ' или ' – апостроф (');
- " или " – кавычка (");
-   или   – неудаляемый пробел;
-
 или
 – переход на новую строку.

В нашем примере правильной будет запись: "3 < 5". Такое содержимое отобразится браузером в виде текста: 3 < 5.

Как уже показано, элементы XML могут включать атрибуты, которые служат для декларации значений свойств элементов или для определения его поведения при отображении документа. Значения атрибутов задаются в виде строк, ограниченных либо кавычками (""), либо апострофами (''). Если в строку, определяющую значение атрибута, необходимо поместить кавычку ("), то строка обрамляется апострофами, если в строке нужен апостроф (') – строка обрамляется кавычками. Если в строке нужны и апостроф, и кавычка – для их представления используют подстановки: (') – для апострофа и (") – для кавычки.

При использовании автономных XAML-файлов применяется правило XML: все стоящие рядом пробельные символы заменяются одним пробелом. Напомним, что к пробельным символам относятся не только пробелы, но и символы табуляции, перехода к началу строки и перехода на следующую строку. Чтобы полностью избежать применения этого правила, применяется атрибут

```
xml:space="preserve"
```

Если нужно только в одном месте XAML-документа в тексте поставить рядом несколько пробелов, то можно вместо названного атрибута использовать для каждого из пробелов эскейп-последовательность « ».

Обратите внимание, что, в отличие от HTML, в записи имен элементов и атрибутов XML учитывается регистр. Существуют еще некоторые особенности XML, которые не влияют на синтаксис XAML, поэтому объяснять их нет необходимости.

Возможность воспроизведения автономных XAML-файлов браузером Internet Explorer поддерживается XAML-службами платформы .NET Framework и сборкой System.XAML. Предоставляемая этими средствами система типов обеспечивает объектно-ориентированное представление языка XAML, при котором каждый элемент XAML-документа отображается на экземпляр соответствующего типа, а каждый атрибут определяет значение свойства этого типа. Таким образом, собственно язык XAML и технология XAML поддерживаются системой типов, которую называют резервной системой типов XAML.

Синтаксический анализатор XAML «проецирует» типы резервной системы типов XAML на систему типов исполняющей среды. В технологиях, реализованных под Windows, исполняющей средой служит CLR (Common Language Runtime), поэтому проецирование выполняется на типы .NET Framework. При использовании другой операционной системы со своей системой типов потребуются другой синтаксический анализатор XAML и другое пространство имен, отличное от вводимого атрибутом

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

До сих пор в примерах каждый XAML-элемент вводился конструкцией, ограниченной парой тегов:

<***имя_элемента*** *атрибуты*>. . . </***имя_элемента***>

Для определения значений свойств XAML-элементов (и соответствующих им объектов резервных типов) в примерах использован традиционный для XML так называемый синтаксис атрибутов. В этом случае каждый атрибут XAML-элемента определял некоторое свойство (или событие) объекта резервного типа и имел вид:

имя_атрибута = "*значение_свойства*"

Атрибуты размещаются при таком синтаксисе в начальном теге XAML-элемента после его имени и отделяются друг от друга пробельными символами. Значение атрибута в этом случае – всегда строка, ограниченная кавычками или апострофами.

Основное различие между XML и XAML состоит в том, что информационная XAML-модель более специализирована, нежели информационная модель XML. Наиболее важным для наших целей отличием является предусмотренная XAML-моделью возможность

представлять XAML-элемент в виде объекта некоторой объектно-ориентированной системы.

В XAML-документе информация может быть представлена в одной из трех форм: объекты, члены (объектов) или текст. Корнем XAML-документа может быть только объект. Объекты могут включать члены, причем включаемые в объект члены не требуется никак упорядочивать. Члены имеют значения. Значением члена может быть либо объект, либо текст. Некоторые члены могут иметь несколько значений. Эти значения одного члена оформляются в виде упорядоченной последовательности (коллекции). В таких последовательностях могут быть одновременно и объекты, и тексты.

На рис. 1.3 показаны фрагмент XAML-документа и графическое представление его информационной XAML-модели.

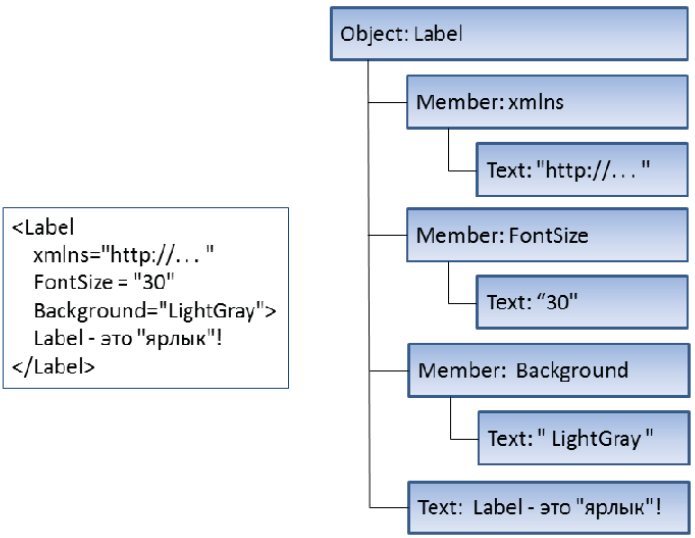

Рис. 1.3. Структура информации XAML-документа

Кроме названного традиционного для XML синтаксиса, при котором свойства элементов определяются значениями атрибутов, XAML позволяет определять свойства, вводя их как элементы XAML-документа. В этом случае говорят о синтаксисе объявления свойств в виде XAML-элементов. Этот способ определения значения свойства применяют в тех случаях, когда синтаксис атрибутов неудобен или не позволяет наглядно объявить свойство.

Формат объявления свойства в виде XAML-элемента:

*<**имя_типа**.имя_свойства>*
Значение_свойства
*</**имя_типа**.имя_свойства>*

Здесь *имя_типа* – имя XAML-элемента, декларирующего объект, для которого определяется свойство. Обратите внимание, что в начальном теге XAML-элемента, декларирующего свойство, отсутствуют (и запрещены!) атрибуты. *Значение_свойства* – либо объект (XAML-элемент), либо текст. При такой декларации свойств говорят (в отличие от синтаксиса атрибутов) о применении синтаксиса XAML-элементов свойств. Таким образом в XAML-разметке различают, и по-разному используют, элементы-объекты и элементы-свойства.

Пример XAML-документа, в котором свойства в декларацию элемента Label вводятся как вложенные XAML-элементы свойств:

```
<!-- №_01_03.xaml - свойства в виде XAML-элементов -->
<Label
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
   <Label.FontSize>30</Label.FontSize>
   <Label.Background>
      LightGray
   </Label.Background>
   <Label.VerticalAlignment>
      Center
   </Label.VerticalAlignment>
   <Label.HorizontalAlignment>
      Center
   </Label.HorizontalAlignment>
   Label - это "ярлык"!
</Label>
```

Этот XAML-документ по результатам его отображения (рис. 1.2) полностью эквивалентен приведенному выше XAML-документу №_01_02.xaml, где свойства определялись атрибутами. Обратите внимание, что при объявлении свойств в виде XAML-элементов их текстовые значения (30, LightGray, Center) записываются без кавычек. Содержимое XAML-элемента Label, определяется, как и ранее, строкой текста, не обрамленного кавычками (Label – это "ярлык"!).

Кроме синтаксиса атрибутов и синтаксиса элементов-свойств, в XAML-разметке существует синтаксис коллекций. Возможность применения синтаксиса коллекций имеется в тех случаях, когда

XAML-элемент является контейнером (то есть его содержимое является коллекцией) или его свойство имеет тип коллекции. Структура XAML-разметки такого элемента:

```
<тип..>
   <тип.коллекция>
      <тип_элемента_коллекции атрибуты/> . . . .
      <тип_элемента_коллекции атрибуты/>
   </тип.коллекция>
</тип..>
```

Кроме такого «полного» объявления коллекций, синтаксис XAML предусматривает возможность объявлять элементы коллекции без обрамления их списка тегами <тип.коллекция>, </тип.коллекция>. В этом случае говорят о неявном синтаксисе коллекций.

Примеры неявного и явного синтаксисов коллекций приведем позже, рассматривая такие контейнеры компоновки, как, например, StackPanel, Canvas, и такие свойства, как Style.Setters и Storyboard.Children.

Пространства имен в XAML-документах

Использование сокращений при записи XAML-«кодов» очень удобно, но иногда создает затруднения для новичков. Мы, например, уже упомянули о возможности объявлять элементы коллекции без обрамления их списка тегами <тип.коллекция>, </тип.коллекция>. Сейчас обратим внимание на применение сокращений при обозначении элементов XAML-документа. До сих пор в примерах использовалось имя XAML-элемента Label без указания того пространства имен, которому это имя принадлежит. Поясним те соглашения, которые обеспечивают эту возможность.

Синтаксический анализатор XAML («парсер» – parser) и/или компилятор, обрабатывая декларацию XAML-документа, для распознавания терминов словаря языка XAML использует то пространство имен, которое определено в XAML-документе атрибутом xmlns. Синтаксис XAML разрешает применять в XAML-документе несколько XAML-словарей, каждый из которых вводится отдельным атрибутом xmlns. Чтобы не было неоднозначности, в название каждого из этих атрибутов может быть добавлен префикс, и название атрибута станет таким:

xmlns:*префикс*

Префикс – это уникальный идентификатор, достаточно произвольно выбранный автором XAML-документа для отнесения использованного в документе имени (элемента, атрибута и т. п.) к соответствующему пространству имен, то есть к конкретному XAML-словарю. Название «префикс» выбрано для этого идентификатора, чтобы подчеркнуть его использование не в обозначении атрибута xmlns, а в названиях тех терминов, которые нужно соотнести с пространством имен, введенным этим атрибутом.

Если в атрибуте, вводящем пространство имен, использовать префикс, то придется и названия XAML-элементов из этого пространства имен в документе записывать с тем же префиксом. Для иллюстрации этого факта перепишем приведенный выше первый автономный XAML-документ (№_01_01.xaml) следующим образом:

```
<!-- №_01_04.xaml - именование пространства Windows -->
<MySpace:Label FontSize = "30" Background="LightGray"
xmlns:MySpace=
"http://schemas.microsoft.com/winfx/2006/xaml/presentation">
   Label - это "ярлык"!
</MySpace:Label>
```

В данном документе в название атрибута xmlns добавлена синтаксическая конструкция «:MySpace» , где MySpace – произвольно выбранное имя. Это имя, используемое в качестве префикса в названии XAML-элемента Label, явным образом указывает его принадлежность пространству имен

```
"http://schemas.microsoft.com/winfx/2006/xaml/presentation".
```

После такого изменения названия атрибута нужно писать в тегах не Label, а только MySpace:Label. При этом отображение XAML-документа не изменится (см. рис. 1.1).

Конечно, явное обозначение принадлежности всех имен к соответствующим им пространствам неудобно, поэтому одно из пространств (а чаще всего то, которое завершается термином «presentation») назначается «главным», не снабжается именем и выбирается по умолчанию для XAML-документов. В этом пространстве имен прописаны названия резервных типов (Label, Button, Page и т. д.), необходимых для обработки XAML-документов в среде ОС Windows (то есть типов из пространств имен System.Windows, System.Windows.Controls, System.Windows.Data, System.Windows.Ink, System.Windows.Media и других).

В качестве второго пространства имен практически все XAML-документы используют пространство, которое вводится атрибутом с префиксом «x» (префикс может быть любым, но достаточно часто используется «x»):

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml".
```

Назначение этого пространства имен – ввести ключевые слова XAML, соответствующие типам из пространства имен System.Windows.Markup. Это пространство имен для языка XAML включает набор обозначений, которые используются особым образом при обработке XAML-документа как компилятором, так и синтаксическим анализатором XAML.

Таблица 1.1. Часто применяемые ключевые слова XAML

Ключевые слова XAML	Описание
x:Array	Позволяет создать в XAML-разметке массив, соответствующий требованиям платформы .NET. Список в XAML-элементе x:Array (представляемый свойством ArrayExtension.Items) содержит элементы массива. Для определения типов элементов массива необходимо применять x:Type
x:Class	Определяет класс корневого элемента XAML-документа
x:Key	Определяет значения ключа для XAML-элемента, помещаемого в словарь
x:Name	Позволяет задать имя XAML-элементу
x:Null	Представляет нулевую (пустую) ссылку
x:Reference	Ссылка на именованный XAML-элемент. Имеет один позиционный параметр, который должен быть именем этого ссылочного элемента
x:Static	Позволяет получить значение статического члена типа (поля, константы или элемента перечисления)
x:Type	Представляет для именованного типа экземпляр класса System.Type

Ключевые слова приведены с префиксом «x:». Именно так они чаще всего используются в XAML-документах.

Еще одно пространство имен (третье) потребуется в тех случаях, когда из XAML-документа потребуется доступ к базовым типам .NET:

```
xmlns:sys="clr-namespace:System;assembly=mscorlib"
```

Именно это пространство имен в начальных версиях XAML поддерживает примитивные типы среды CLR (Common Language Runtime). То есть для типа Int32 (тип целочисленных значений ординарной длины) используется обозначение sys:Int32.

В спецификации XAML 2009 для обозначения базовых (BuiltIn) типов .NET Framework введены следующие ключевые слова:

- x:Boolean (логический тип, в C# bool);
- x:Byte (однобайтовый знаковый тип, в C# byte);
- x:Char (символьный тип, в C# char);
- x:Decimal (десятичный тип, в C# decimal);
- x:Double (вещественный тип двойной точности, в C# double);
- x:Int16 (короткий целочисленный тип, в C# short);
- x:Int32 (ординарный целочисленный тип, в C# int);
- x:Int64 (целочисленный тип двойной точности, в C# long);
- x:Object (базовый тип .NET Framework, в C# object);
- x:Single (вещественный тип одинарной точности, в C# float);
- x:String (тип, в C# string).

Кроме того, в пространство имен XAML спецификация XAML 2009 помещает типы:

- x:Dictionary;
- x:List;
- x:TimeSpan;
- x:Uri.

В разных областях применениях XAML (Silverlight, WPF, WinRT, UWP и Xamarin.Forms) используются и другие пространства имен, которые нам пока не понадобятся.

Обработка XAML-документов

Приводя примеры автономных XAML-файлов, мы никак не объясняли, почему и как их обрабатывает браузер. Теперь еще раз обратим внимание на расширение «.xaml» в имени XAML-файла. Именно это расширение позволяет браузеру обрабатывать текст

документа как код XAML. Если открыть браузер Internet Explorer и ввести полный адрес файла, например №_01_02.xaml, то в центре окна браузера (см. рис. 1.2) появится поле метки (ярлыка) с надписью «Label – это "ярлык"!». Тот же эффект будет, если в Проводнике дважды щелкнуть мышью по имени файла или запустить файл №_01_02.xaml как программу из командной строки.

Те средства, которые в OS Windows (и в частном случае в браузере Internet Explorer) позволяют обрабатывать такие автономные XAML-файлы, можно подробно не рассматривать. Достаточно знать, что в современных версиях Windows эти средства присутствуют и ими можно пользоваться. С их помощью на основе XAML-кода создается объект класса Page (производного от класса FrameworkElement), пригодный для отображения в веб-браузере. Эти же средства формируют объекты для элементов XAML-документа (в нашем примере создается объект класса Lable). Затем на основе XAML-кода для этого объекта определяется текстовое значение свойства Content («Label – это "ярлык"!»). А атрибуты в начальном теге (или XAML-элементы свойств) определяют значения свойств FontSize, VerticalAlignment, HorizontalAlignment, Background.

Чтобы объяснить, какова роль объекта класса Page и откуда «появились» свойства объекта класса Label, перечислим общие принципы технологии XAML.

1. Набор элементов XAML определен совокупностью поддерживающих эту технологию классов библиотеки платформы .NET, которые в Windows служат резервной системой типов XAML.
2. Имена элементов XAML не могут быть произвольными, а полностью совпадают с именами резервных типов (поддерживающих их классов библиотеки).
3. С помощью атрибутов XAML-элементов задаются свойства объектов, соответствующих этим элементам.
4. Атрибуты элементов XAML обозначаются именами членов (свойств, событий) классов, соответствующих этим элементам.
5. XAML-документ всегда представляет собой дерево вложенных элементов, причем в любом XAML-документе только один корневой элемент.
6. В любом XAML-элементе только один вложенный XAML-элемент, то есть элемент содержимого.

7. Для некоторых XAML-элементов содержимое является коллекцией (например, списком), причем отдельный элемент этой коллекции может быть либо текстом, либо XAML-элементом.
8. При использовании XAML в технологиях среды выполнения под Windows в качестве корневого элемента дерева XAML-документа применяют объекты одного из классов: Window, Page, Application, NavigationWindow, ResourceDictionary.
9. При обработке браузером автономного XAML-файла (или XAML-файла, включенного в приложение, например WPF, UWP или SilverLight) его элементы используются для генерации объектов, формирующих интерфейс пользователя.

Итак, корневой элемент XML-файла всегда один. По умолчанию корневой элемент *автономного* XAML-документа становится содержимым для элемента класса Page, который автоматически создается при обработке XAML-кода и отображается веб-браузером. Если в XAML-документе явно определить в качестве корневого элемент класса Page, то новый элемент Page по умолчанию не создается, а отображается тот элемент Page, который явно определен в XAML-документе. В этом случае появляется возможность задать свойства не только таких вложенных элементов, как в нашем примере Label, но и самого элемента Page.

Дополним предыдущий пример автономного XAML-файла (№_01_02.xaml) корневым элементом Page.

```
<!-- №_01_05.xaml - явная декларация элемента Page -->
<Page
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      Width = "200" Height = "150" Background="Gray">
   <Label  FontSize = "30"
      VerticalAlignment = "Center"
      HorizontalAlignment = "Center"
      Background="LightGray">
      Label - это "ярлык"!
   </Label>
</Page>
```

В этом примере атрибут xmlns, определяющий пространство имен XAML, перенесен из элемента Lable в явно определенный корневой элемент Page. Для корневого элемента Page дополнительно определены атрибуты: Width = "200" Height = "150" Background="Gray". Теперь размеры отображаемой браузером страницы (Page) фиксированы, и при заданном размере шриф-

та ярлык (Lable) не может целиком отобразиться (см. рис. 1.4). Связано это с тем фактом, что по умолчанию размеры прямоугольника ярлыка определяются размерами строки выводимого в нем текста, а эти размеры (ширина прямоугольника) больше явно определенной ширины страницы (Width = "200"). В предыдущем примере (см. рис. 1.2) размеры прямоугольника ярлыка были те же, но ширина страницы не была фиксированной и автоматически устанавливалась так, чтобы изображение ярлыка было полностью видимым.

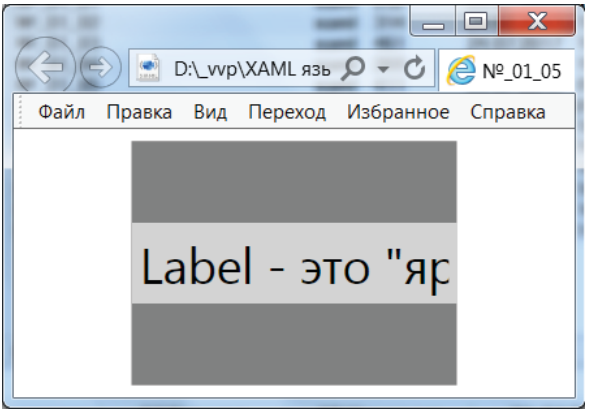

Рис. 1.4. Представление автономного XAML-файла в окне браузера

> **Примечание.** Корневым элементом *автономного XAML-файла*, отображаемого браузером, нельзя сделать ни элемент Window, ни элемент Application. Роли этих корневых элементов важны в других приложениях.

XAML и резервные типы .NET

До сих пор в примерах использовались всего два XAML-элемента: ярлык Lable и корневой элемент Page. Как-то стихийно, без особых объяснений, в объявлениях этих элементов появились атрибуты, причем выбор их был мало обоснован. Для дальнейшего объяснения синтаксиса XAML (и особенно для иллюстрации его особенностей) полезно ввести еще несколько элементов и хотя бы перечислить наиболее важные из их атрибутов.