

WPF (Windows Presentation Foundation)

WPF – новая технология .NET Framework 3.x для создания пользовательских интерфейсов в клиентских приложениях.

Одна из задач, которая была решена при проектировании и реализации WPF, - разделение работы между дизайнерами и программистами.

Решение состоит в разделении исходного кода WPF-приложения на две части:

- декларативное описание пользовательского интерфейса с использованием языка разметки XAML (EXtensible Application Markup Language);
- код на языке программирования, например C#, содержащий обработку событий.

Для компиляции WPF-приложений обычно используется Microsoft Build Engine (MSBuild) –технология, включенная в .NET Framework 3.x в виде набора сборок.

Компиляция WPF приложений

- ✓ При компиляции XAML файлы разбираются (parsed) и преобразуются в коды на языке BAML (Binary Application Markup Language), которые встраиваются как ресурс в исполняемый файл.
- ✓ Код BAML компактнее исходного XAML кода и при выполнении его загрузка выполняется быстрее, чем загрузка XAML файла.
- ✓ При компиляции для каждого XAML файла создается файл с кодом на языке программирования, содержащий частичное (partial) объявление класса для элемента верхнего уровня в файле разметки.
- ✓ В общем случае компиляция XAML файлов осуществляется в два прохода.
- ✓ Сначала компилируются только те XAML файлы, которые не содержат ссылки на локально-определенные типы (т.е. типы, определенные где-нибудь в данном проекте), так как они существуют только в виде исходного кода и еще не были скомпилированы.
- ✓ XAML файлы со ссылками на локально-определенные типы компилируются при втором проходе компилятора.
- ✓ Вся необходимая для работы MSBuild информация об исходных файлах, ссылках на зависимые сборки и конфигурация приложения находится в файлах проекта MSBuild - XML файлах, подчиняющихся MSBuild schema.

XML (EXtensible Markup Language)

- ✓ XML – простой гибкий текстовый формат, который используется как основа для создания языков разметки для публикации документов и обмена данными.
- ✓ Стандарт XML издан в виде рекомендаций консорциума всемирной паутины (World Wide Web Consortium - W3C) – международной организации, которая занимается технологическими стандартами для всемирной паутины.
- ✓ Документ XML состоит из элементов.
- ✓ Элементы документа могут быть вложенными, но не могут пересекаться. Таким образом, элементы документа образуют дерево.
- ✓ Каждый документ XML должен иметь один и только один корневой элемент.
- ✓ Элементы могут содержать атрибуты , представляющие собой пары имя-значение.

Элементы XML

- ✓ Каждый элемент документа должен иметь имя.
- ✓ Имя элемента - это строка символов, которая начинается с подчеркивания или буквы, и состоит только из букв, цифр, символов подчеркивания (underscore), дефисов (hyphen) и точек (period). Имена элементов чувствительны к регистру.
- ✓ Каждый элемент начинается с открывающего тэга и заканчивается закрывающим тэгом. Все, что находится между открывающим и закрывающим тэгом, называется содержимым (content) элемента.

- ✓ Открывающий тэг имеет вид

`< elementName >`

или (для элемента с атрибутами)

`<elementName att1Name="att1Value" att2Name="att2Value">`

Закрывающий тэг имеет вид `</ elementName >` .

- ✓ Элемент может быть пустым. Пустой элемент может содержать атрибуты. Для пустого элемента допустима форма с пустым тэгом (empty tag)

`<elementName att1Name="att1Value" att2Name="att2Value"/>`

Данные элемента XML

- ✓ Текстовые данные XML-элемента представляют собой либо символьные данные (CDATA), либо анализируемые символьные данные, либо их комбинацию.
- ✓ Можно вставить символы Unicode в виде кодов символа в десятичном или шестнадцатиричном формате (character references), используя синтаксис

<code>& #value;</code>	десятичный формат (например, символ Евро <code>&#8364;</code>)
<code>&# xvalue;</code>	шестнадцатиричный формат (например, символ Евро <code>&#x20AC;</code>)

- ✓ Определены пять подстановочных строк (entity references), которые можно использовать в самом документе вместо символов угловых скобок `<>` и символа слэш `/`, чтобы XML-анализатор не трактовал их как управляющие символы языка разметки.

<code>&lt;</code>	<code><</code> (меньше)
<code>&gt;</code>	<code>></code> (больше)
<code>&amp;</code>	<code>&</code> (амперсанд)
<code>&apos;</code>	<code>'</code> (апостроф или одинарная кавычка)
<code>&quot;</code>	<code>"</code> (двойные кавычки)

Разделы CDATA

- ✓ Все символы, которые находятся между комбинацией символов `<![CDATA[` и `]]>` трактуются XML анализатором как содержимое, не включающее символов разметки. Все остальные данные – это анализируемые символьные данные, которые XML-анализаторы интерпретируют как язык разметки.
- ✓ Подстановочные строки (character references) не работают внутри разделов CDATA.
- ✓ Разделы CDATA не могут быть вложенными.

Комментарии

- ✓ Комментарии – любая текстовая информация, размещенная между символами `<!--` и `-->`.
- ✓ Комментарии можно разместить внутри пролога документа, в содержимом документа или после документа.
- ✓ Комментарии не могут появиться внутри значений атрибутов документа или внутри тэгов.

Атрибуты XML

- ✓ Элемент может иметь атрибуты. Атрибуты состоят из пар имя-значение.
`<elementName att1Name="att1Value" att2Name="att2Value" ... </elementName >`
- ✓ Имена атрибутов подчиняются тем же требованиям, что и имена элементов.
- ✓ Атрибуты размещаются в открывающем тэге элемента. Число атрибутов элемента не ограничено. Пустой элемент может содержать атрибуты.
- ✓ Значение атрибута заключается в одинаковые кавычки – простые или двойные. Для разных атрибутов можно использовать разные кавычки. Внутри двойных кавычек можно использовать одинарные без использования `"`; и наоборот.
- ✓ Атрибуты должны иметь как имя, так и значение.
- ✓ Внутри атрибутов можно использовать `entity references` и `character references`, но нельзя использовать разделы `CDATA`.
- ✓ Порядок определения атрибутов может не сохраняться XML-анализатором.
- ✓ XML-анализаторы не сохраняют число пробелов в значениях атрибутов.

Пространства имен XML

- ✓ Чтобы избежать конфликтов при использовании элементов с одинаковыми именами, в документе можно определить пространства имен.
- ✓ Пространство имен определяется с помощью атрибута `xmlns` с префиксом или без него.
- ✓ Для пространств имен обычно используются URI (Uniform Resource Identifier). URI не обязан ссылаться на реально существующий объект.
- ✓ Пространство имен по умолчанию задается атрибутом `xmlns` без префикса. Элемент, в котором определено пространство имен по умолчанию, и все его дочерние элементы автоматически относятся к этому пространству имен, если явно не указано другое пространство имен.
- ✓ Объявление пространства имен с префиксом имеет вид `xmlns:имя=URI`
- ✓ Имя задает префикс пространства имен.
- ✓ Префикс можно использовать как в том элементе, в котором он определен, так и в его дочерних. Атрибут не помещается автоматически в то же пространство имен, что и содержащий его элемент.
- ✓ Пространство имен действует только в пределах того элемента, атрибутом которого является его декларация.

Схема документа XML

- ✓ Схема документа (schema) определяет допустимую структуру документа – формальные спецификации имен элементов документа, определяющие допустимые имена элементов и их комбинации, отношения дочерний-родительский между элементами документа , последовательность, в которых могут появиться дочерние элементы, и их число.
- ✓ Для схемы документа используется один из языков описания структуры XML документа.
- ✓ Консорциум W3C рекомендует использовать как стандарт - XML Schema Definition (XSD) – язык на основе XML для описания схем XML-документов.
- ✓ Файл, содержащий XML Schema, обычно имеет расширение «.xsd» (XML-schema-definition).
- ✓ Документ называется well-formed, если он корректен с точки зрения синтаксиса XML.
- ✓ Документ называется valid, если его структура согласуется со схемой документа.

Модель данных документа XML

- ✓ После проверки документа на соответствие XML Schema программа может создать модель данных документа, включающую
 - словарь (названия элементов и атрибутов) ;
 - модель содержания (отношения между элементами и атрибутами и их структура) ;
 - типы данных.
- ✓ Каждый элемент в этой модели ассоциируется с определённым типом данных, позволяя строить в памяти объект, соответствующий структуре XML-документа.

XAML (EXtensible Application Markup Language)

- ✓ XAML – построенный на базе XML язык разметки для описания объектов .NET Framework.
- ✓ Подмножество WPF XAML языка используется в технологии WPF для описания пользовательского интерфейса.
- ✓ XAML – декларативный язык разметки и дает возможность отделить описание объектов пользовательского интерфейса от логики выполнения приложения (run-time logic) .
- ✓ Конструкция частичного определения класса дает возможность описать класс в нескольких файлах – исполняемый код в файлах с исходным кодом, например, на C#, а декларативную часть класса – на языке разметки XAML.
- ✓ В XAML определен набор правил, по которым элементам ставятся в соответствие классы или структуры, атрибутам – свойства или события, а пространства имен XML отображаются на пространства имен CLR.

Пространства имен XAML

- ✓ Два пространства имен – пространство имен XAML и пространство имен WPF - присутствуют во всех WPF документах:

```
<Window x:Class="Wpf_Abc.Window1"
  xmlns= "http://schemas.microsoft.com/winfx/2006/xaml/presentation " <!--пространство имен WPF-->
  xmlns:x= "http://schemas.microsoft.com/winfx/2006/xaml " <!--пространство имен XAML-->
  Title="Window1" Height="300" Width="300">
  <Grid>
    <Button Height="23" Margin="90,0,114,50" Name="button1" VerticalAlignment="Bottom"
Click="button1_Click">Button Click</Button>
  </Grid>
</Window>
```

- ✓ Синтаксический анализатор использует атрибут Class для создания класса, производного от типа, совпадающего именем элемента (Window в данном примере) .
- ✓ Атрибут Class указан с префиксом x: , что означает, что это имя находится в пространстве имен XAML.