Учебные задания 1

Создание XML-документа Технология LINQ to XML.

Учебные задания 1 .

Во всех заданиях данной подгруппы предполагается, что исходные текстовые файлы содержат текст в кодировке «windows-1251», а все файловые строки являются непустыми. Создаваемые XML-документы также должны иметь кодировку «windows-1251». Узлом (node) XML-документа называется любой его компонент, не являющийся атрибутом (attribute), в частности, комментарий (comment), инструкция обработки (processing instruction), обычный текст. Элементом (element) XMLдокумента называется именованный компонент, который может содержать другие узлы, а также иметь атрибуты. В объектной модели X-DOM, входящей в состав интерфейса LINQ to XML, с каждым видом компонентов XML-документа связан соответствующий класс: XObject общий предок всех компонентов, XNode общий предок всех узлов, XText текстовый узел, т. е. узел, представляющий собой обычный текст, XComment комментарий, XProcessingInstruction инструкция обработки, XElement элемент, XAttribute атрибут. С XML-документом связан класс XDocument. Если некоторый узел B XML-документа содержится внутри некоторого элемента A, то элемент A называется предком (ancestor) узла B, а узел B потомком (descendant) элемента A. Если элемент A является ближайшим предком узла B, то B называется дочерним узлом (child node) элемента A, а элемент A родительским элементом (parent) узла B; если при этом узел B является элементом, то он называется дочерним элементом (child element) элемента A. Первый элемент XML-документа называется корневым элементом (root); корневой элемент является предком для всех других элементов XML-документа, сам корневой элемент предков не имеет. Корневой элемент считается элементом нулевого уровня, его дочерние узлы/элементы узлами/элементами первого уровня, их дочерние узлы/элементы узлами/элементами второго уровня, и т. д. В XML-документе имеется единственный элемент нулевого уровня (корневой элемент), однако могут присутствовать несколько узлов нулевого уровня (комментариев или инструкций обработки). Если в задании говорится, что элемент содержит текстовую строку (или число), то это означает, что соответствующая строка (или строковое представление числа) является дочерним текстовым узлом данного элемента. Во всех заданиях предполагается, что элемент имеет не больше одного дочернего текстового узла, а текстовый узел содержит хотя бы один значащий символ (т. е. символ, отличный от пробела и управляющих символов). При сохранении XML-документа следует использовать метод Save класса XDocument с единственным параметром именем файла; это обеспечит автоматическое форматирование сохраняемого документа и удаление всех текстовых узлов, не содержащих значащих символов. Элементы, не содержащие дочерних узлов, могут представляться в двух вариантах: в виде парных тегов, между которыми отсутствует текст (<a></a>), и в виде одного комбинированного тега (<a />). Элементы, не содержащие узлов, могут иметь атрибуты. Если в условии сказано, что дан XML-документ, то это означает, что дано имя файла, содержащего этот документ. Преобразование XML-документа всегда должно завершаться сохранением преобразованного документа в том же файле, из которого был считан исходный вариант этого документа. Если в условии упоминаются порядковые номера элементов некоторой последовательности, то предполагается, что нумерация начинается от 1. В заданиях подгрупп, предшествующих подгруппе «Работа с пространствами имен XML-документа», предполагается, что имена всех элементов и атрибутов XML-документа имеют пустое пространство имен. Указания, приведенные к некоторым заданиям, следует учитывать и при выполнении последующих заданий текущей подгруппы.

LinqXml1. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом root и элементами первого уровня line, каждый из которых содержит одну строку из исходного файла. Указание. В конструкторе корневого элемента использовать последовательность объектов XElement, полученную методом Select из исходного набора строк.

LinqXml2. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом root и элементами первого уровня, каждый из которых содержит одну строку из исходного файла и имеет имя line с приписанным к нему порядковым номером строки (line1, line2 и т. д.).

LinqXml3. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом root и элементами первого уровня line, каждый из которых содержит одну строку из исходного файла. Элемент, содержащий строку с порядковым номером N (1, 2, ), должен иметь атрибут num со значением, равным N.

LinqXml4. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) слов, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом root, элементами первого уровня line и элементами второго уровня word. Элементы line соответствуют строкам исходного файла и не содержат дочерних текстовых узлов, элементы word каждого элемента line содержат по одному слову из соответствующей строки (слова располагаются в алфавитном порядке).

LinqXml5. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) слов, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом root, элементами первого уровня line и элементами второго уровня word. Элементы line соответствуют строкам исходного файла и не содержат дочерних текстовых узлов, элементы word каждого элемента line содержат по одному слову из соответствующей строки (слова располагаются в порядке их следования в исходной строке). Элемент line должен содержать атрибут num, равный порядковому номеру строки в исходном файле, элемент word должен содержать атрибут num, равный порядковому номеру слова в строке.

LinqXml6. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) целых чисел, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом root, элементами первого уровня line и элементами второго уровня number. Элементы line соответствуют строкам исходного файла и не содержат дочерних текстовых узлов, элементы number каждого элемента line содержат по одному числу из соответствующей строки (числа располагаются в порядке убывания). Элемент line должен содержать атрибут sum, равный сумме всех чисел из соответствующей строки.

**Учебные задания 2**

LinqXml7. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) целых чисел, в документ. Порядок имен атрибутов должен соответствовать порядку их первого вхождения в документ. Указание. Использовать методы SelectMany и Distinct. разделенных ровно одним пробелом. Создать XML-документ LinqXml14. Дан XML-документ. Найти элементы второго с корневым элементом root, элементами первого уровня уровня, имеющие дочерний текстовый узел, и вывести количество найденных элементов, а также имя каждого найден- line и элементами второго уровня sum-positive и number-negative. Элементы line соответствуют строкам исходного файла и не содержат дочерних текстовых узлов, элемент sum-positive является первым дочерним ного элемента и значение его дочернего текстового узла. Порядок вывода элементов должен соответствовать порядку их следования в документе. элементом каждого элемента line и содержит сумму всех LinqXml15. Дан XML-документ, содержащий хотя бы один положительных чисел из соответствующей строки, элементы элемент первого уровня. Для каждого элемента первого number-negative содержат по одному отрицательному числу из соответствующей строки (числа располагаются в порядке, обратном порядку их следования в исходной строке). уровня найти количество его потомков, имеющих не менее двух атрибутов, и вывести имя элемента первого уровня и найденное количество его потомков. Элементы выводить в алфавитном порядке их имен, а при совпадении имен в порядке возрастания найденного количества потомков. LinqXml8. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) слов, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом root, элементами первого уровня line, элементами второго уровня word и элементами третьего уровня char. Элементы line и word не содержат дочерних текстовых узлов. Элементы line соответствуют строкам исходного файла, элементы word каждого элемента line соответствуют словам из этой строки (слова располагаются в алфавитном порядке), элементы char каждого элемента word содержат по одному символу из соответствующего слова (символы располагаются в порядке их следования в слове). LinqXml9. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом root, элементами первого уровня line и комментариями (комментарии являются дочерними узлами корневого элемента). Если строка текстового файла начинается с текста «comment:», то она (без текста «comment:») добавляется в XML-документ в качестве очередного комментария, в противном случае строка добавляется в качестве дочернего текстового узла в очередной элемент line. LinqXml10. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом root, элементами первого уровня line и инструкциями обработки (инструкции обработки являются дочерними узлами корневого элемента). Если строка текстового файла начинается с текста «data:», то она (без текста «data:») добавляется в XML-документ в качестве данных к очередной инструкции обработки с именем instr, в противном случае строка добавляется в качестве дочернего текстового узла в очередной элемент line. Анализ содержимого XML-документа LinqXml11. Дан XML-документ. Найти все различные имена его элементов и вывести каждое найденное имя вместе с числом его вхождений в документ. Имена элементов выводить в порядке их первого вхождения. Указание. Использовать метод GroupBy. LinqXml12. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Найти все различные имена элементов первого уровня и вывести каждое найденное имя вместе с числом его вхождений в документ в качестве имени элемента первого уровня. Имена элементов выводить в алфавитном порядке. LinqXml13. Дан XML-документ, содержащий хотя бы один атрибут. Вывести все различные имена атрибутов, входящих LinqXml16. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Для каждого элемента первого уровня найти суммарное количество атрибутов у его элементов-потомков второго уровня (т. е. элементов, являющихся дочерними элементами его дочерних элементов) и вывести найденное количество атрибутов и имя элемента. Элементы выводить в порядке убывания найденного количества атрибутов, а при совпадении количества атрибутов в алфавитном порядке имен. LinqXml17. Дан XML-документ, содержащий хотя бы один текстовый узел. Найти все различные имена элементов, имеющих дочерний текстовый узел, и вывести эти имена, а также значения всех связанных с ними дочерних текстовых узлов. Имена выводить в алфавитном порядке; текстовые значения, связанные с каждым именем, выводить в порядке их появления в документе. LinqXml18. Дан XML-документ, содержащий хотя бы один атрибут. Найти все различные имена атрибутов и вывести эти имена, а также все связанные с ними значения (все значения считаются текстовыми). Порядок имен должен соответствовать порядку их первого вхождения в документ; значения, связанные с каждым именем, выводить в алфавитном порядке. LinqXml19. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Для каждого элемента первого уровня найти его дочерние элементы, имеющие максимальное количество потомков (при подсчете числа потомков учитывать также потомки-узлы, не являющиеся элементами). Перебирая элементы первого уровня в порядке их появления в XML-документе, вывести имя элемента, число N максимальное количество потомков, имеющихся у его дочерних элементов (значение N может быть равно 0), и имена всех дочерних элементов, имеющих N потомков (имена дочерних элементов выводить в алфавитном порядке; среди этих имен могут быть совпадающие). Если элемент первого уровня не содержит дочерних элементов, то в качестве значения N выводить 1, а в качестве имени дочернего элемента текст «no child». LinqXml20. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Для каждого элемента первого уровня найти его элементы-потомки, имеющие максимальное количество атрибутов. Перебирая элементы первого уровня в порядке их появления в XML-документе, вывести имя элемента, число N максимальное количество атрибутов у его потомков (значение N может быть равно 0) и имена потомков, имеющих N атрибутов (имена потомков выводить в алфавитном порядке; среди этих имен могут быть совпадающие). Если элемент первого уровня не содержит элементов-потомков, то в качестве значения N выводить 1, а в качестве имени потомка текст «no child».