Преобразование XML-текста в дерево объектов DOM

Чтобы обработка XML-документа посредством ActionScript стала возможной, его нужно преобразовать в дерево объектов DOM. Для этого соответствующий текст должен быть передан XML-анализатору ActionScript. Сделать это можно тремя способами:

• Чаще всего XML-текст разбирается на этапе создания объекта класса XML. Как вы помните, этот объект описывает документ в целом (в отличие от класса XMLNode, объекты которого исполняют роль узлов дерева DOM), и через него осуществляются все манипуляции с данными. Чтобы связать объект класса XML сразу же при его создании с XML-документом, строка с текстом последнего должна быть передана конструктору класса XML:

```
var xml_doc:String="<doc><link>http://www.piter.com</link></doc>"; var

xml_data:XML=new XML(xml_doc);
```

• Если один и тот же объект класса XML должен по мере проигрывания фильма описывать разные XML-документы, то нужно использовать метод parseXML(). Данный метод принимает текст XML-документа в качестве параметра, передает его анализатору, а затем связывает вызвавший его объект класса XML с полученным деревом DOM. Старое содержимое объекта класса XML при этом уничтожается.

```
var xml_doc:String="<doc><link>http://www.piter.com</link></doc>"; var

xml_data:XML=new XML("<doc0></doc0>");

// Заменяем дерево DOM одного документа деревом другого
xml_data.parseXML(xml_doc);
```

• Предыдущие два способа преобразования XML-документа в дерево DOM подразумевают, что строка с ним уже имеется в фильме. Но зачастую XML-документ, подлежащий разбору, хранится в удаленном текстовом файле. Поэтому перед разбором он должен быть импортирован. Дать команду на закачку файла с документом позволяет метод load() класса XML. Когда документ будет закачан, он будет проанализирован, а полученное дерево DOM будет связано с вызвавшим load() объектом. Старое его содержимое при этом будет удалено. Так как заведомо неизвестно, когда будет получен файл с XML- документом, метод load() обычно используется в сочетании с событием onData (происходит при поступлении данных) или onLoad (возникает по завершении разбора XML-документа). По особенностям использования метод load() и события onData и onXML абсолютно идентичны одноименным элементам уже разобранного нами в предыдущей статье класса LoadVars.

```
var xml_data:XML = new XML();

xml_data.load("http://www.mysite.ru/data.xml");

xml_data.onLoad = function():Void {

trace("XML-документ закачан и разобран");

};
```

Вне зависимости от того, как текст XML-документа поступает анализатору, разбирается он по одним и тем же принципам. Основные из них мы обсудим ниже.

У XML, в отличие от HTML, очень строгие правила синтаксиса. Поэтому допустить ошибку, которая приведет к прекращению разбора документа анализатором, просто. К счастью, обнаружить, что поступивший на обработку XML-документ некорректен и на основании него не было сформировано дерева DOM, несложно. С учетом строгости

синтаксиса XML, точный “диагноз” причин сбоя может поставить сам анализатор XML Flash-плеера. Узнать, был ли разбор XML-документа успешным, и если нет, то почему, позволяет свойство status класса XML. Данное свойства равняется 0, если преобразование XML-текста в дерево DOM прошло без накладок. При возникновении сбоя status будет хранить отрицательное число от –1 до –10. Каждое число соответствует отдельному типу синтаксической ошибки. То, какое число какую ошибку означает, показано в таблице 1.

Таблица 1. Коды синтаксических ошибок

Код	Описание
0	Документ синтаксически корректен: ни одной ошибки не обнаружено
-2	Раздел CDATA неверно завершен (скорее всего, введена одна “]” вмес
-3	Неверно завершено объявление XML-документа (возможно, для задан
-4	Неверно задано объявление типа документа
-5	Неверно задан комментарий
-6	Неверно сформирован узел XML-документа
-7	Свободной оперативной памяти недостаточно для размещения дерева
-8	Неверно задан атрибут (скорее всего, его значение не было взято в ка
-9	Задан начальный тег, но нет конечного
-10	Есть конечный тег, но нет начального

При изучении таблицы 1 не может не возникнуть следующего вопроса: если в XML-документе имеется несколько разноплановых ошибок, сообщение о какой из них будет помещено в свойство status? Ответ: первой от начала документа. Дело в том, что синтаксический анализатор работает до тех пор, пока ему не попадется ошибка. Текст документа, лежащий за ней, не анализируется. Однако _____(это важно знать) дерево объектов DOM на основании разобранной части документа формируется даже если в нем встречается синтаксическая ошибка.

Приведем небольшой пример. Передадим анализатору Flash-плеера документ, в котором атрибут тега не помещен в кавычки:

```
var xml_data:XML = new XML("<doc><data type=xml></data></doc>");
```

```
trace(xml_data.status); // Выводит: -6
```

В общем, синтаксическую ошибку анализатор нашел — но не совсем точно. Он сообщил, что узел задан неверно. Более информативным было бы значение свойства status, равное –8. Применяйте свойство status, если XML-документов, которые будет использовать фильм, нет на момент его создания. Это позволит сделать приложение более устойчивым к сбоям. XML-анализатор Flash-плеера проверяет документ на синтаксическую корректность. Однако он не осуществляет проверки на действительность. То есть, если в документе есть ссылка на DTD, которому он должен соответствовать, анализатор плеера не будет сравнивать документ с описанным в DTD шаблоном. Он даже не будет импортировать DTD. Однако объявление типа документа не будет просто выброшено. Оно будет занесено в свойство docTypeDecl в форме строки.

Пример:

```
var docTDecl:String="<!DOCTYPE data SYSTEM 'dtd/data.dtd'>";

var xml_body:String=docTDecl+"<data></data>";

var xml_doc:XML=new XML(xml_body);

trace(xml_doc.docTypeDecl);// Выводит: <!DOCTYPE data SYSTEM 'dtd/data.dtd'>
```

Аналогично объявлению типа документа, анализатор XML Flash-плеера не осуществляет разбора объявления XML-документа. Из этого можно сделать несколько важных выводов:

• Если XML-документ создается специально для передачи данных фильму Flash, XML-объявление будет почти наверняка излишним.

• Невозможно в фильм передать XML-документ, кодировка которого отлична от используемой фильмом, просто указав ее в атрибуте encoding XML-объявления. В любом случае фильм будет трактовать документ исходя из той кодировки, которую он применяет сам.

Несмотря на то, что XML-заголовок никак не влияет на результат преобразования документа в дерево DOM, он не отбрасывается, а помещается в специальное свойство xmlDecl как строка.

Пример:

```
var doc:String="<?xml version='1.0' encoding='UTF-8'?>";

doc+="<data></data>"

var xml_doc:XML=new XML(doc);

trace(xml_doc.xmlDecl); // Выводит: <?xml version='1.0' encoding='UTF-8'?>
```

Попробуйте определить, сколько узлов и какого уровня будет иметь описывающее следующий XML-документ дерево DOM:

```
<data> </data>
```

“Один узел первого уровня”, — скажете вы. И окажетесь не правы. При описании DOM, используемой Flash-плеером, мы говорили о том, что текст, вложенный в тег, дает особый текстовый узел. “Но между тегами <data> нет текста”, — возразите вы. Да, видимого текста действительно нет. Но есть пробел. А он является таким же символом,

как и любой другой. Следовательно, для его хранения будет создан текстовый узел. То же самое произойдет, если между тегов имеется перенос строки, перевод каретки или символ табуляции. То, что совершенно любой текст, набранный в теге, преобразуется в текстовый узел, не всегда приемлемо. Например, если вы создаете XML-документ вручную, то для его лучшей читабельности стоит использовать отступы и переносы. Однако при этом в дереве DOM появится масса новых текстовых узлов, которые с высокой вероятностью собьют алгоритм “с толку”. Чтобы этого не произошло, нужно перед передачей текста документа анализатору присвоить значение true свойству ignoreWhite. При этом если текст тега состоит из одних лишь пробельных символов, он будет попросту отброшен. Повторные пробелы между словами, а также вначале и в конце текста удаляться не будут. По умолчанию свойство ignoreWhite равняется false.

Пример:

```
var xml_obj:XML = new XML("<text> </text>");

trace(xml_obj.firstChild.firstChild.nodeValue == " ");

// Выводит: true // ( текстовый узел на основании пробела был создан)

xml_obj.ignoreWhite = true;

xml_obj.parseXML("<text> </text>");

trace(xml_obj.firstChild.firstChild); // Выводит: null (узла нет)
```

Анализатор XML Flash-плеера может распознавать секции CDATA. Эту его возможность нужно использовать, если XML-документ содержит другой XML-документ или, что чаще встречается на практике, HTML-текст. Содержимое секции CDATA не анализируется, а без каких-либо изменений преобразуется в текстовый узел.

Например:

```
var xml_text:String="<text><![CDATA[<b>Это <i>HTML</i>-текст</b>]]></text>";

var xml_obj:XML=new XML(xml_text);

trace(xml_obj.firstChild.firstChild.nodeValue); // Выводит: <b>Это <i>HTML</i>-текст</b>
```