

文章

姚鑫 · 六月 26, 2021 阅读大约需 6 分钟

第十九章 使用%XML.TextReader

第十九章 使用%XML.TextReader

%XML.TextReader类提供了一种简单、容易的方法来读取可能直接映射到InterSystems IRIS对象，也可能不直接映射到InterSystems IRIS对象的任意XML文档。具体地说，该类提供了导航格式良好的XML文档并查看其中信息(元素、属性、注释、名称空间URI等)的方法。该类还基于DTD或XML架构提供完整的文档验证。但是，与%XML.Reader不同的是，%XML.TextReader不提供返回DOM的方法。如果需要DOM，请参阅前面的“将XML导入对象”一章。

注意：使用的任何XML文档的XML声明都应该指明该文档的字符编码，并且文档应该按照声明的方式进行编码。如果未声明字符编码，InterSystems IRIS将使用前面的“输入和输出的字符编码”中描述的默认值。如果这些默认值不正确，请修改XML声明，使其指定实际使用的字符集。

创建文本阅读器Text Reader方法

要读取不一定与IRIS对象类有任何关系的任意XML文档，可以调用%XML.TextReader类的方法，该类将打开文档并将其作为文本阅读器对象加载到临时存储中。文本阅读器对象包含一个可导航的节点树，每个节点都包含有关源文档的信息。然后，方法可以导航该文档并查找有关该文档的信息。对象的属性提供有关文档的信息，这些信息取决于在文档中的当前位置。如果存在验证错误，这些错误也可以作为树中的节点使用。

整体结构

法应执行以下部分或全部操作：

1. 通过以下方法之一的第一个参数指定文档源：

Method	First Argument
ParseFile()	文件名，带有完整路径。请注意，文件名和路径只能包含ASCII字符。
ParseStream()	流
ParseString()	字符串
ParseURL()	URL

在任何情况下，源文档都必须是格式良好的XML文档；也就是说，它必须遵守XML语法的基本规则。这些方法中的每一个都返回一个状态(\$OK或失败代码)，以指示结果是否成功。可以使用常用机制测试状态；特别是可以使用\$System.Status.DisplayError(status)查看错误消息的文本。

对于这些方法中的每一个，如果该方法返回\$OK，则它通过引用(其第二个参数)返回包含XML文档中的信息的文本阅读器对象。

其他参数允许控制实体解析、验证、找到哪些项等。这些内容将在本章后面的“解析方法的参数列表”中介绍。

2. 检查解析方法返回的状态，并在适当的情况下退出。

如果解析方法返回\$OK，则有一个与源XML文档相对应的文本阅读器对象。可以导航此对象。

文档可能包含 “element”、“endelement”、“startprefixmapping” 等节点。

重要提示:在任何验证错误的情况下,文档包含 “错误” 或 “警告” 节点。
代码应该检查这些节点。

3. 使用以下实例方法之一开始读取文档。

- 使用Read()导航到文档的第一个节点。
- 使用ReadStartElement()导航到特定类型的第一个元素。
- 使用MoveToContent()导航到类型为 “chars” 的第一个节点。

4. 获取该节点感兴趣的属性的值(如果有的话)。可用的属性包括名称、值、深度等。

5. 根据需要继续在文档中导航并获取属性值。

如果当前节点是元素,则可以使用MoveToAttributeIndex()或MoveToAttributeName()方法将焦点移至该元素的属性。若要返回到元素(如果适用),请使用MoveToElement()。

6. 如果需要,可以使用Rewind()方法返回到文档的开头(第一个节点之前)。这是唯一可以在源代码中倒退的方法。

方法运行后,文本读取器对象将被销毁,所有相关的临时存储都将被清除。

示例1

下面是一个简单的方法,它可以读取任何XML文件,并显示每个节点的序列号、类型、名称和值:

```
/// w ##class(PHA.TEST.Xml).WriteNodes("E:\temp\textReader.txt")
ClassMethod WriteNodes(myfile As %String)
{
    set status = ##class(%XML.TextReader).ParseFile(myfile,.textreader)
    //????
    if $$$ISERR(status) {do $System.Status.DisplayError(status) quit}
    //????????
    while textreader.Read()
    {
        Write !, "Node ", textreader.seq, " is a(n) "
        Write textreader.NodeType, " "
        If textreader.Name="" {
            Write "named: ", textreader.Name
        } Else {
            Write "and has no name"
        }
        Write !, "    path: ",textreader.Path
        If textreader.Value="" {
            Write !, "    value: ", textreader.Value
        }
    }
    q ""
}
```

此示例执行以下操作:

1. 它调用ParseFile()类方法。这将读取源文件，创建一个文本阅读器对象，并通过引用在变量doc中返回该对象。
2. 如果ParseFile()成功，则该方法然后调用read()方法来查找文档中的每个后续节点。
3. 对于每个节点，该方法写入包含节点序列号、节点类型、节点名称(如果有)、节点路径和节点值(如果有)的输出行。输出将写入当前设备。

以下示例源文档：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="mystyles.css"?>
<Root>
  <s01:Person xmlns:s01="http://www.root.org">
    <Name attr="yx">yaoxin</Name>
    <DOB>1990-04-25</DOB>
  </s01:Person>
</Root>
```

对于此源文档，前面的方法生成以下输出：

```
DHC-APP>w ##class(PHA.TEST.Xml).WriteNodes("E:\temp\textReader.txt")
```

```
Node 1 is a(n) processinginstruction named: xml-stylesheet
  path:
  value: type="text/css" href="mystyles.css"
Node 2 is a(n) element named: Root
  path: /Root
Node 3 is a(n) startprefixmapping named: s01
  path: /Root
  value: s01 http://www.root.org
Node 4 is a(n) element named: s01:Person
  path: /Root/s01:Person
Node 5 is a(n) element named: Name
  path: /Root/s01:Person/Name
Node 6 is a(n) chars and has no name
  path: /Root/s01:Person/Name
  value: yaoxin
Node 7 is a(n) endelement named: Name
  path: /Root/s01:Person/Name
Node 8 is a(n) element named: DOB
  path: /Root/s01:Person/DOB
Node 9 is a(n) chars and has no name
  path: /Root/s01:Person/DOB
  value: 1990-04-25
Node 10 is a(n) endelement named: DOB
  path: /Root/s01:Person/DOB
Node 11 is a(n) endelement named: s01:Person
  path: /Root/s01:Person
Node 12 is a(n) endprefixmapping named: s01
  path: /Root
  value: s01
Node 13 is a(n) endelement named: Root
  path: /Root
```

请注意，注释已被忽略；默认情况下，%XML.TextReader忽略注释。

Example 2

下面的示例读取一个XML文件并列出其中的每个元素

```
/// w ##class(PHA.TEST.Xml).ShowElements("E:\temp\textReader.txt")
ClassMethod ShowElements(myfile As %String)
{
    set status = ##class(%XML.TextReader).ParseFile(myfile, .textreader)
    if $$$ISERR(status) {do $System.Status.DisplayError(status) quit}
    while textreader.Read()
    {
        if (textreader.NodeType = "element")
        {
            write textreader.Name,!
        }
    }
    q ""
}
```

此方法使用NodeType属性检查每个节点的类型。如果节点是元素，则该方法将其名称打印到当前设备。对于前面显示的XML源文档，此方法生成以下输出：

```
DHC-APP>w ##class(PHA.TEST.Xml).ShowElements("E:\temp\textReader.txt")
Root
s01:Person
Name
DOB
```

节点类型

文档的每个节点都是以下类型之一：

文本阅读器文档中的节点类型

Type	Description
"attribute"	XML属性。
"chars"	一组字符(如元素的内容)。%XML.TextReader类识别其他节点类型(“CDATA”、“EntityReference”和“EndEntity”)，但自动将它们转换为“字符”。
"comment"	XML注释。
"element"	XML元素的开始。
"endelement"	XML元素的结束。
"endprefixmapping"	声明名称空间的上下文的结束。
"entity"	XML实体。
"error"	解析器发现的验证错误。
"ignorablewhitespace"	混合内容模型中标记之间的空白。
"processinginstruction"	XML处理指令。
"startprefixmapping"	XML命名空间声明，它可能包括也可能不包括命名空间。
"warning"	解析器发现验证警告。

请注意，XML元素由多个节点组成。例如，以下XML片段：

```
<Person>
  <Name>Willeke, Clint B.</Name>
  <DOB>1925-10-01</DOB>
```

</Person>

SAX解析器将此XML视为以下节点集：

文档节点示例

Node Number	Type of Node	Name of Node, If Any	Value of Node, If Any
1	element	Person	
2	element	Name	
3	chars		Willeke, Clint B.
4	endelement	Name	
5	element	DOB	
6	chars		1925-10-01
7	endelement	DOB	
8	endelement	Person	

例如，注意<DOB>元素被认为是三个节点：一个元素节点、一个字符节点和一个结束元素节点。还要注意，该元素的内容只能作为chars节点的值使用。

[#Caché](#)

源

URL:

<https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%B9%9D%E7%AB%A0-%E4%BD%BF%E7%94%A8xmltextreader>