文章
姚 鑫 · 六月 15, 2021 阅读大约需 5 分钟

第七章 控制命名空间分配的外观

第七章 控制命名空间分配的外观

控制命名空间分配的外观

除了控制命名空间分配外，还可以控制命名空间分配在XML输出中的显示方式。具体地说，可以控制以下内容：

显式名称空间分配与隐式名称空间分配

将元素和属性分配给命名空间时，XML中有两种等效的表示形式，由编写器实例的SuppressXmlns属性控制。

为一个名为Person的对象生成XML输出，该对象被分配给名称空间 “ http://www.person.org ” (通过前面讨论的namespace类参数)。
使用缺省输出(SuppressXmlns等于0)的示例如下:

```
<Person xmlns="http://www.person.com" GroupID="4">
  <Name>Uberoth,Amanda Q.</Name>
  <DOB>1952-01-13</DOB>
</Person>
```

另一种可能的形式完全相同，如下所示。
这是使用SuppressXmlns等于1生成的，它确保显式分配给名称空间的每个元素都显示为该名称空间的前缀。

```
<s01:Person xmlns:s01="http://www.person.com" GroupID="4">
  <s01:Name>Uberoth,Amanda Q.</s01:Name>
  <s01:DOB>1952-01-13</s01:DOB>
</s01:Person>
```

请注意，此属性仅影响命名空间分配的显示方式；它不控制如何分配任何命名空间。如果不使用命名空间，则此参数无效。

为命名空间指定自定义前缀

当为对象生成XML输出时，系统会根据需要生成命名空间前缀。第一个名称空间前缀是s01，下一个是s02，依此类推。可以指定不同的前缀。为此，请在启用XML的对象本身的类定义中设置XMLPREFIX参数。此参数有两个效果：
- 它确保在XML输出中声明指定的前缀。也就是说，即使没有必要这样做，它也会被声明。
- 它使用该前缀，而不是在其他情况下会看到的自动生成的前缀。

控制空字符串("")的导出方式

为对象启用XML时，需要指定将空值和空字符串投影到XML的方式

其中一个选项是在支持xml的类中将XMLIGNORENULL设置为 “ RUNTIME ” (不区分大小写)。
在这种情况下，当使用%XML.Write的RuntimeIgnoreNull属性的值来确定如何处理任何等于""的属性，如下所示:

- **如果编写器的RuntimeIgnoreNull属性为0(默认值)，则XMLNIL参数控制如何导出该属性。XMLNIL是一个类参数和一个属性参数；属性参数优先。**
    - 如果XMLNIL为0(默认值)，则不投影特性。也就是说，它不包含在XML文档中。
    - 如果XMLNIL为1，并且该属性用作元素，则该属性将按如下方式导出：

```
<PropName xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
```

- 如果XMLNIL为1并且特性用作属性，则不会输出特性。
    - 如果编写器的RuntimeIgnoreNull属性为1，则该属性将导出为空元素或空属性(其导出方式与值$char(0)相同，后者始终导出为空元素或空导出)。

除非XMLIGNORENULL在启用xml的类中是 “ RUNTIME ”，否则编写器的RuntimeIgnoreNull属性是无效的。

示例:RuntimeIgnoreNull为0(默认值)

```
Class EmptyStrings.Export Extends (%Persistent, %XML.Adaptor)
{

Parameter XMLNAME="Test";


Parameter XMLIGNORENULL = "RUNTIME";


///?????????
///XMLNIL is 0, the default
Property Property1 As %String;


///?????????
///XMLNIL is 0, the default
Property Property2 As %String(XMLPROJECTION = "ATTRIBUTE");


///????XMLNIL=1???
Property Property3 As %String(XMLNIL = 1);


///????XMLNIL=1???????
Property Property4 As %String(XMLNIL=1,XMLPROJECTION="ATTRIBUTE");

}
```

如果创建了这个类的一个新实例(并且没有设置任何属性的值)，然后使用 %XML.Writer输出，如下所示:

```
<?xml version="1.0" encoding="UTF-8"?>
<Test>
  <Property3 xsi:nil="true"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
</Test>
```

例如:RuntimeIgnoreNull为1

```
<?xml version="1.0" encoding="UTF-8"?>
<Test Property2="" Property4="">
  <Property1></Property1>
  <Property3></Property3>
</Test>
```

在本例中，因为RuntimeIgnoreNull为1，所以没有使用XMLNIL参数。
相反，""被导出为空属性或空元素。

导出类型信息

默认情况下，XML编写器不写入类型信息。有两个选项可用于在输出中包括类型信息：

- 编写器的OutputTypeAttribute属性。如果此属性为1，则编写器包括其写入的对象内所有元素的XML类型信息(但不包括对象本身)。例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Person>
    <Name xsi:type="s:string">Petersburg,Greta U.</Name>
    <DOB xsi:type="s:date">1949-05-15</DOB>
  </Person>
</Root>
```

请注意，相应的命名空间将添加到XML文档的根。

- Object()和RootObject()方法的className参数。此参数用于指定对象的预期ObjectScript类型(类名)。

如果参数与实际类型相同，则编写器不包括对象的类型信息。

如果参数与实际类型不同，编写器将包括对象的实际XML类型(默认为类名)。例如，假设Test2.PersonWithAddress实例编写输出，并将className参数指定为MyPackage.MyClass。因为MyPackage.MyClass与实际的类名不同，所以编写器生成以下输出：

```
<?xml version="1.0" encoding="UTF-8"?>
<PersonWithAddress xsi:type="PersonWithAddress"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Name>Avery,Robert H.</Name>
  <Address>
    <City>Ukiah</City>
    <Zip>82281</Zip>
  </Address>
</PersonWithAddress>
```

生成SOAP编码的XML

对于%XML.Writer，Format属性控制输出的整体格式。这是以下选项之一：

- “ literal ”，即默认值，在本书的大多数例子中都使用了它。
- “ encoded ”，按照SOAP 1.1标准中的描述进行编码。
- “ encoded12 ”，按照SOAP 1.2标准中的描述进行编码。

创建内联引用

在编码格式中，任何对象值属性都被作为引用包含，被引用的对象被导出为单独的元素。

要将这些属性内联导出，而不是作为单独的元素，请将ReferencesInline属性设置为1。

如果格式是 “ literal ”， ReferencesInline属性没有效果。

导出后控制unswizling

当导出一个支持xml的持久对象时，系统会像往常一样自动将所有需要的信息混合到内存中;该信息包括对象值属性。
导出对象后，InterSystems IRIS将消除任何对象列表，但(默认情况下)不会消除单个对象引用。
对于大对象，这可能导致<STORE>错误。

在这种情况下，要使任何单个对象引用不被混合，请在支持xml的类中设置XMLUNSWIZZLE参数，如下所示:

```
Parameter XMLUNSWIZZLE = 1;
```

该参数默认值为0。

控制元素的关闭

只包含属性的元素可以用以下任一方式表示：

```
<myelementname attribute="value" attribute="value" attribute="value"></myelementname>
<myelementname attribute="value" attribute="value" attribute="value"/>
```

Object()方法始终使用第一个语法导出元素。如果需要使用此处显示的第二种语法关闭元素，请手动编写对象，如本章前面的 “ 手动构造元素 ” 中所述。

#Caché

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%B8%83%E7%AB%A0-%E6%8E%A7%E5%88%B6%E5%91%BD%E5%90%8D%E7%A9%BA%E9%97%B4%E5%88%86%E9%85%8D%E7%9A%84%E5%A4%96%E8%A7%82