文章
姚 鑫 · 六月 11, 2021 阅读大约需 4 分钟

第四章 添加命名空间声明

第四章 添加命名空间声明

添加命名空间声明

默认行为

在%XML.Writer会自动插入命名空间声明，生成命名空间前缀，并在适当的地方应用前缀。例如，以下类定义：

```
Class Sample.Person Extends (%Persistent, %Populate, %XML.Adaptor)
{

Parameter NAMESPACE = "http://www.yaoxin.com";
}
```

如果导出此类的多个对象，则会看到类似以下内容：

```
DHC-APP> w ##class(Demo.XmlDemo).Obj2Xml(1)
<?xml version="1.0" encoding="UTF-8"?>
<Person xmlns="http://www.yaoxin.com">
  <Name>yaoxin</Name>
  <SSN>111-11-1117</SSN>
  <DOB>1990-04-25</DOB>
  <s01:Home xmlns="" xmlns:s01="http://www.yaoxin.com">
    <Street>889 Clinton Drive</Street>
    <City>St Louis</City>
    <State>WI</State>
    <Zip>78672</Zip>
  </s01:Home>
  <s01:Office xmlns="" xmlns:s01="http://www.yaoxin.com">
    <Street>9619 Ash Avenue</Street>
    <City>Ukiah</City>
    <State>AL</State>
    <Zip>56589</Zip>
  </s01:Office>
  <Spouse>
    <Name>???</Name>
    <SSN>111-11-1115</SSN>
    <FavoriteColors>
      <FavoriteColorsItem>Red</FavoriteColorsItem>
      <FavoriteColorsItem>Orange</FavoriteColorsItem>
      <FavoriteColorsItem>Yellow</FavoriteColorsItem>
      <FavoriteColorsItem>Green</FavoriteColorsItem>
    </FavoriteColors>
  </Spouse>
  <FavoriteColors>
```

```
    <FavoriteColorsItem>Red</FavoriteColorsItem>
    <FavoriteColorsItem>Orange</FavoriteColorsItem>
    <FavoriteColorsItem>Yellow</FavoriteColorsItem>
  </FavoriteColors>
  <Age>31</Age>
</Person>
```

名称空间声明会自动添加到每个<Person> 元素。只将其添加到文档的根目录。

手动添加声明

可以控制何时将命名空间引入XML输出。以下方法都会影响所写入的下一个元素(但不会影响该元素之后的任何元素)。为方便起见，其中几种方法添加了标准的W3名称空间。

通常使用这些方法将命名空间声明添加到文档的根元素；也就是说，在调用RootObject()或RootElement()之前调用其中一个或多个方法。

注意：这些方法都没有将任何元素分配给名称空间，并且这些名称空间永远不会作为默认名称空间添加。在生成特定元素时，需要指明它使用的名称空间，如后面的 “ 编写根元素 ” 和 “ 生成XML元素 ” 中所述。

AddNamespace()

```
method AddNamespace(namespace As %String,
                    prefix As %String,
                    schemaLocation As %String) as %Status
```

添加指定的命名空间。这里，Namespace是要添加的名称空间，Prefix是该名称空间的可选前缀，schemaLocation是指示相应架构位置的可选URI。

如果未指定前缀，则会自动生成前缀(格式为S01、S02等)。

下面的示例显示了此方法的效果。首先，假设Person类被分配给一个名称空间(类参数中的NAMESPACE)。如果在未首先调用AddNamespace()方法的情况下生成此类实例的输出，则可能会收到如下所示的输出：

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
  <Person xmlns="http://www.person.org">
    <Name>Love,Bart Y.</Name>
...
```

或者，在编写根元素之前按如下方式调用AddNamespace()方法：

```
 set status=writer.AddNamespace("http:///www.person.org","p")
```

如果随后生成根元素，则输出如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns:p="http:///www.person.org">
  <Person xmlns="http://www.person.org">
...
```

或者，假设在调用AddNamespace()方法时指定了第三个参数，该参数提供了关联架构的位置：

```
 set status=writer.AddNamespace("http:///www.person.org","p","http://www.MyCompany.co
m/schemas/person.xsd")
```

在这种情况下，如果随后生成Root元素，则输出如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns:p="http:///www.person.org"
xsi:schemaLocation="http:///www.person.org http://www.MyCompany.com/schemas/person.xs
d"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Person xmlns="http://www.person.org">
...
```

AddInstanceNamespace()

```
method AddInstanceNamespace(prefix As %String) as %Status
```

添加W3架构实例命名空间。这里的前缀是用于此命名空间的可选前缀。默认前缀为XSI。

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

AddSchemaNamespace()

```
method AddSchemaNamespace(prefix As %String) as %Status
```

添加W3架构命名空间。这里的前缀是用于此命名空间的可选前缀。默认前缀为s。

```
<Root xmlns:s="http://www.w3.org/2001/XMLSchema">
...
```

AddSOAPNamespace()

```
method AddSOAPNamespace(soapPrefix As %String,
                        schemaPrefix As %String,
                        xsiPrefix As %String) as %Status
```

添加W3 SOAP编码命名空间、SOAP架构命名空间和SOAP架构实例命名空间。此方法有三个可选参数：用于这些命名空间的前缀。默认前缀分别为SOAP-Enc、s和XSI。

```
<Root xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
```

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

AddSOAP12Namespace()

```
method AddSOAP12Namespace(soapPrefix As %String,
                          schemaPrefix As %String,
                          xsiPrefix As %String) as %Status
```

添加W3 SOAP 1.2编码命名空间、SOAP架构命名空间和SOAP架构实例命名空间。

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
```

可以使用这些方法中的多个方法。如果使用其中的多个命名空间，则受影响的元素将包含所有指定命名空间的声明。

编写根元素

每个XML文档必须恰好包含一个根元素。有两种方法可以创建此元素：

- 根元素可能直接对应于一个启用了InterSystems IRIS XML的对象。

在本例中，使用RootObject()方法，该方法将指定的启用XML的对象作为根元素写入。输出包括该对象中包含的所有对象引用。根元素获取该对象的结构，不能插入其他元素您可以指定根元素的名称，也可以使用由启用XML的对象定义的默认值。

前面的示例使用了此技术。

- 根元素可能只是一组元素的包装器(可能是一组支持XML的对象)。

在本例中，使用RootElement()方法，该方法插入具有指定名称的根级元素。如果此文档缩进，此方法还会增加后续操作的缩进级别。

然后调用其他方法为根元素内的一个或多个元素生成输出。在根目录中，可以按照选择的任何顺序或逻辑包含所需的元素。之后，调用EndRootElement()方法关闭根元素。

在这两种情况下，都可以指定要用于根元素的命名空间，只有在启用了XML的类没有Namespace参数值的情况下才会应用该命名空间。

请记住，如果文档包含文档类型声明，则该DTD的名称必须与根元素的名称相同。

#Caché #InterSystems IRIS

源URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E5%9B%9B%E7%AB%A0-%E6%B7%BB%E5%8A%A0%E5%91%BD%E5%90%8D%E7%A9%BA%E9%97%B4%E5%A3%B0%E6%98%8E