文章
姚 鑫 · 七月 5, 2021 阅读大约需 7 分钟

第二十八章 定制SAX解析器创建自定义内容处理程序

[toc]

第二十八章 定制SAX解析器创建自定义内容处理程序

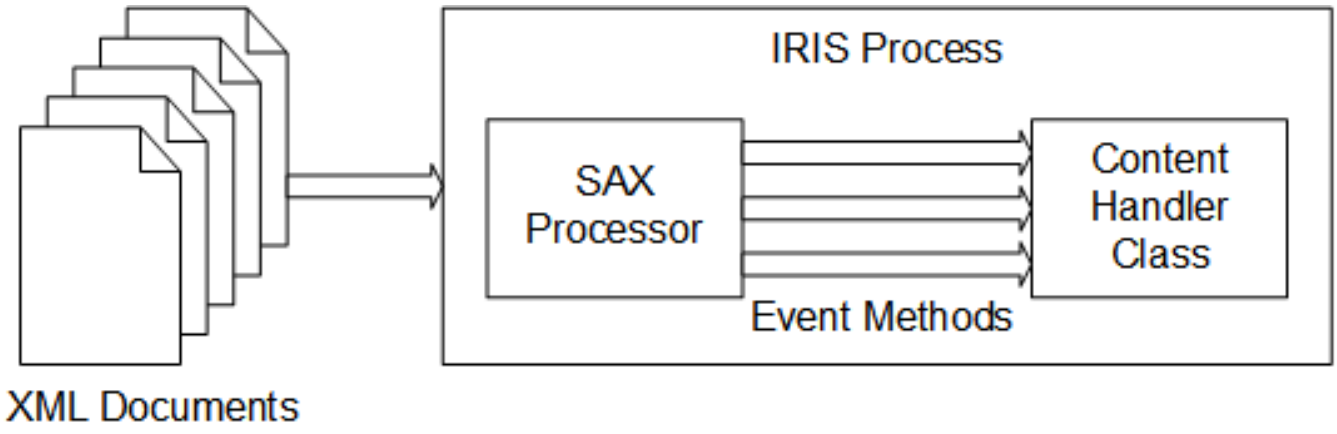
创建自定义内容处理程序

如果直接调用InterSystems IRIS
SAX解析器，则可以根据自己的需要创建自定义内容处理程序。本节讨论以下主题：
- Overview
- 要在内容处理程序中自定义的方法的描述
- %XML.SAX.Parser类中解析方法的参数列表摘要
- 示例

创建自定义内容处理程序概述

要定制InterSystems IRIS SAX解析器导入和处理XML的方式，请创建并使用定制的SAX内容处理程序。具体地说，创建%XML.SAX.ContentHandler的子类。然后，在新类中，重写任何默认方法以执行所需的操作。在解析XML文档时使用新的内容处理程序作为参数；为此，需要使用%XML.SAX.Parser类的解析方法。

此操作如下图所示：



创建和使用自定义导入机制的过程如下：

1. 创建扩展%XML.SAX.ContentHandler的类。
2. 在该类中，包括希望覆盖的方法，并根据需要提供新定义。
3. 在使用%XML.SAX.Parser的分析方法之一(即ParseFile()、ParseStream()、ParseString()或ParseURL())编写读取XML文档的类方法。

调用分析方法时，请将自定义内容处理程序指定为参数。

SAX内容处理程序的可定制方法

%XML.SAX.ContentHandler类在特定时间自动执行某些方法。通过覆盖它们，您可以自定义内容处理程序的行为。

响应事件

%XML.SAX.ContentHandle类分析XML文件，并在它到达XML文件中的特定点时生成事件。根据事件的不同，会执行不同的方法。这些方法如下：

- OnPostParse() — 在XML解析完成时触发。
- characters() — 由字符数据触发。
- comment() — 注释触发
- endCData() —由CDATA部分的末尾触发。
- endDocument() —由文档结尾触发。
- endDTD() — 由DTD结束触发。
- endElement() —由元素的末尾触发。
- endEntity() — 由一个实体的终结触发。
- endPrefixMapping() — 由名称空间前缀映射的结束触发。
- ignorableWhitespace() — 由元素内容中的可忽略空格触发。
- processingInstruction() — 由XML处理指令触发。
- skippedEntity() — 被跳过的实体触发。
- startCData() —由CDATA部分的开头触发。
- startDocument() — 由文档的开头触发。
- startDTD() — 由DTD的开头触发。
- startElement() — 由元素的开始触发。
- startEntity() — 由一个实体的开始触发。
- startPrefixMapping() — 由名称空间前缀映射的开始触发。

默认情况下，这些方法是空的，可以在自定义内容处理程序中覆盖它们。

处理错误

%XML.SAX.ContentHandler类在遇到某些错误时也会执行方法：

- error() — 由可恢复的解析器错误触发。
- fatalError() — 由致命的XML解析错误触发。
- warning() — 由解析器警告通知触发。

默认情况下，这些方法为空，可以在自定义内容处理程序中重写它们。

计算事件掩码

当调用InterSystems IRIS SAX解析器(通过%XML.SAX.Parser类)时，可以指定一个掩码参数来指示哪些回调是感兴趣的。如果未指定掩码参数，解析器将调用内容处理程序的Mask()方法。此方法返回一个整数，该整数指定与内容处理程序的重写方法相对应的复合掩码。

例如，假设创建了一个自定义内容处理程序，其中包含startElement()和endElement()方法的新版本。在本例中，Mask()方法返回一个数值，该数值等于$$$SAXSTARTELEMENT和$$$SAXENDELEMENT,之和，这两个标志对应于这两个事件。如果没有为解析方法指定掩码参数，则解析器将调用内容处理程序的Mask()方法，因此只处理这两个事件。

其他有用的方法

%XML.SAX.ContentHandler类提供在特殊情况下有用的其他方法：

- LocatePosition()-通过引用返回两个参数，这两个参数指示解析的文档中的当前位置。第一个表示行号，第二个表示行偏移。
- PushHandler()-在堆栈上推送新的内容处理程序。SAX的所有后续回调都将转到这个新的内容处理程序，直到该处理程序完成处理。

如果在解析一种类型的文档时遇到想要以不同方式解析的一段XML，则可以使用此方法。在本例中，当检测到要以不同方式处理的段时，调用PushHandler()方法，该方法将创建一个新的内容处理程序实例。所有回调都会转到此内容处理程序，直到调用PopHandler()返回上一个内容处理程序。
- PopHandler()-返回堆栈上的上一个内容处理程序。

这些是final方法，不能重写。

SAX解析方法的参数列表

要指定文档源，请使用%XML.SAX.Parser类的ParseFile()、ParseStream()、ParseString()或ParseURL()方法。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。完整的参数列表按顺序如下：

1. pFilename, pStream, pString, or pURL — 文档源.
2. pHandler — 内容处理程序，它是%XML.SAX.ContentHandler类的实例。
3. pResolver — 分析源时使用的实体解析器。
4. pFlags — 用于控制SAX解析器执行的验证和处理的标志。
5. pMask — 用于指定XML源中感兴趣的项的掩码。通常不需要指定此参数，因为对于%XML.SAX.Parser的解析方法，默认掩码为0。这意味着解析器调用内容处理程序的Mask()方法。该方法通过检测(在编译期间)在事件处理程序中自定义的所有事件回调来计算掩码。只处理那些事件回调。
6. pSchemaSpec —
   验证文档源所依据的架构规范。此参数是一个字符串，其中包含以逗号分隔的命名空间/URL对列表：

```
"namespace URL,namespace URL"
```

这里，Namespace是用于模式的XML名称空间，URL是提供模式文档位置的URL。名称空间和URL值之间有一个空格字符。

7. pHttpRequest (For the ParseURL() method only) — 这里，Namespace是用于模式的XML名称空间，URL是提供模式文档位置的URL。名称空间和URL值之间有一个空格字符。
8. pSSLConfiguration — 客户端SSL/TLS配置的配置名称。

注意：请注意，此参数列表与%XML.TextReader类的解析方法略有不同。有一点不同，%XML.TextReader不提供指定自定义内容处理程序的选项。

SAX处理程序示例

想要一个文件中出现的所有XML元素的列表。要做到这一点，只需记录每个开始元素。那么这个过程是这样的：

1. 创建一个名为MyApp.Handler的类，它扩展%XML.SAX.ContentHandler：

```
Class MyApp.Handler Extends %XML.SAX.ContentHandler
{
}
```

2. 使用以下内容覆盖startElement()方法：

```
Class MyApp.MyHandler extends %XML.SAX.ContentHandler
{
// ...

Method startElement(uri as %String, localname as %String,
              qname as %String, attrs as %List)
{
    //we have found an element
    write !,"Element: ",localname
}

}
```

3. 将一个类方法添加到读取和分析外部文件的Handler类：

```
Class MyApp.MyHandler extends %XML.SAX.ContentHandler
{
// ...
ClassMethod ReadFile(file as %String) as %Status
{
    //create an instance of this class
    set handler=..%New()

    //parse the given file using this instance
    set status=##class(%XML.SAX.Parser).ParseFile(file,handler)

    //quit with status
    quit status
}
}
```

请注意，这是一个类方法，因为它在应用程序中被调用以执行其处理。此方法执行以下操作：

1. 它创建内容处理程序对象的实例：

```
set handler=..%New()
```

2. 它在一个调用%XML.SAX.Parser的ParseFile()方法。这将验证并解析文档(由fileName指定)，并调用内容处理程序对象的各种事件处理方法：

```
    set status=##class(%XML.SAX.Parser).ParseFile(file,handler)
```

每次在解析器解析文档时发生事件(如开始或结束元素)时，解析器都会调用内容处理程序对象中的适当方法。在本例中，唯一被覆盖的方法是startElement()，它随后写出元素名称。对于其他事件，例如到达End元素，不会发生任何事情(默认行为)。

3. 当ParseFile()方法到达文件末尾时，它返回。处理程序对象超出作用域，并自动从内存中删除。

4. 在应用程序中的相应点，调用ReadFile（）方法，将文件传递给解析：

```
 Do ##class(Samples.MyHandler).ReadFile(filename)
```

其中，filename是正在读取的文件的路径。

例如，如果文件的内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
  <Person>
    <Name>Edwards,Angela U.</Name>
    <DOB>1980-04-19</DOB>
    <GroupID>K8134</GroupID>
    <HomeAddress>
      <City>Vail</City>
      <Zip>94059</Zip>
    </HomeAddress>
    <Doctors>
      <Doctor>
        <Name>Uberoth,Wilma I.</Name>
      </Doctor>
      <Doctor>
        <Name>Wells,George H.</Name>
      </Doctor>
    </Doctors>
  </Person>
</Root>
```

则此示例的输出如下所示：

```
Element: Root
Element: Person
Element: Name
Element: DOB
Element: GroupID
Element: HomeAddress
Element: City
Element: Zip
Element: Doctors
Element: Doctor
Element: Name
Element: Doctor
Element: Name
```

使用HTTPS

%XML.SAX.Parser支持HTTPS。也就是说，可以使用此类执行以下操作：
- (对于ParseURL())解析HTTPS位置提供的XML文档。
- (对于所有解析方法)解析HTTPS位置的实体。

在所有情况下，如果这些项目中的任何一个是在HTTPS位置上提供的，请执行以下操作:

1. 使用管理门户创建包含所需连接详细信息的SSL/TLS配置。这是一次性的步骤。
2. 调用%XML.SAX.Parser的适用解析方法时，请指定pSSLConfiguration参数。

默认情况下，InterSystems IRIS使用Xerces图元解析。%XML.SAX.Parser仅在以下情况下使用其自己的实体解析：
- PSSLConfiguration参数非空。
- 已配置代理服务器。

#Ensemble

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%BA%8C%E5%8D%81%E5%85%AB%E7%AB%A0-%E5%AE%9A%E5%88%B6sax%E8%A7%A3%E6%9E%90%E5%99%A8%E5%88%9B%E5%BB%BA%E8%87%AA%E5%AE%9A%E4%B9%89%E5%86%85%E5%AE%B9%E5%A4%84%E7%90%86%E7%A8%8B%E5%BA%8F