← Ruby Socket 编程 Ruby Web Service 应用 – SOAP4R →

Ruby XML, XSLT 和 XPath 教程

什么是 XML？

XML 指可扩展标记语言（eXtensible Markup Language）。

可扩展标记语言，标准通用标记语言的子集，一种用于标记电子文件使其具有结构性的标记语言。

它可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。 它非常适合万维网传输，提供统一的方法来描述和交换独立于应用程序或供应商的结构化数据。

更多内容请查看我们的 XML 教程

XML解析器结构和API

XML的解析器主要有DOM和SAX两种。

- SAX解析器是基于事件处理的，需要从头到尾把XML文档扫描一遍，在扫描的过程中，每次遇到一个语法结构时，就会调用这个特定语法结构的事件处理程序，向应用程序发送一个事件。
- DOM是文档对象模型解析，构建文档的分层语法结构，在内存中建立DOM树，DOM树的节点以对象的形式来标识，文档解析文成以后，文档的整个DOM树都会放在内存中。

Ruby 中解析及创建 XML

RUBY中对XML的文档的解析可以使用这个库REXML库。

REXML库是ruby的一个XML工具包，是使用纯Ruby语言编写的，遵守XML1.0规范。

在Ruby1.8版本及其以后，RUBY标准库中将包含REXML。

REXML库的路径是： rexml/document

所有的方法和类都被封装到一个REXML模块内。

REXML解析器比其他的解析器有以下优点：

- 100% 由 Ruby 编写。
- 可适用于 SAX 和 DOM 解析器。
- 它是轻量级的,不到2000行代码。
- 很容易理解的方法和类。
- 基于 SAX2 API 和完整的 XPath 支持。
- 使用 Ruby 安装，而无需单独安装。

以下为实例的 XML 代码，保存为movies.xml:

```
<collection shelf="New Arrivals">
<movie title="Enemy Behind">
<type>War, Thriller</type>
```

```
<format>DVD</format>
<year>2003</year>
<rating>PG</rating>
<stars>10</stars>
<description>Talk about a US-Japan war</description>
</movie>
<movie title="Transformers">
<type>Anime, Science Fiction</type>
<format>DVD</format>
<year>1989</year>
<rating>R</rating>
<stars>8</stars>
<description>A schientific fiction</description>
</movie>
<movie title="Trigun">
<type>Anime, Action</type>
<format>DVD</format>
<episodes>4</episodes>
<rating>PG</rating>
<stars>10</stars>
<description>Vash the Stampede!</description>
</movie>
<movie title="Ishtar">
<type>Comedy</type>
<format>VHS</format>
<rating>PG</rating>
<stars>2</stars>
<description>Viewable boredom</description>
</movie>
</collection>
```

DOM 解析器

让我们先来解析 XML 数据，首先我们先引入 rexml/document 库，通常我们可以将 REXML 在顶级的命名空间中引入：

实例

```
#!/usr/bin/ruby -w
require 'rexml/document'
include REXML
xmlfile = File.new("movies.xml")
xmldoc = Document.new(xmlfile)
# 获取 root 元素
root = xmldoc.root
puts "Root element : " + root.attributes["shelf"]
# 以下将输出电影标题
xmldoc.elements.each("collection/movie"){
|e| puts "Movie Title : " + e.attributes["title"]
}
# 以下将输出所有电影类型
xmldoc.elements.each("collection/movie/type") {
|e| puts "Movie Type : " + e.text
}
# 以下将输出所有电影描述
```

```
xmldoc.elements.each("collection/movie/description") {
|e| puts "Movie Description : " + e.text
}
```

以上实例输出结果为：

```
Root element : New Arrivals
Movie Title : Enemy Behind
Movie Title : Transformers
Movie Title : Trigun
Movie Title : Ishtar
Movie Type : War, Thriller
Movie Type : Anime, Science Fiction
Movie Type : Anime, Action
Movie Type : Comedy
Movie Description : Talk about a US-Japan war
Movie Description : A schientific fiction
Movie Description : Vash the Stampede!
Movie Description : Viewable boredom
SAX-like Parsing:
```

SAX 解析器

处理相同的数据文件：movies.xml，不建议SAX的解析为一个小文件，以下是个简单的实例：

实例

```
#!/usr/bin/ruby -w
require 'rexml/document'
require 'rexml/streamlistener'
include REXML
class MyListener
include REXML::StreamListener
def tag_start(*args)
puts "tag_start: #{args.map {|x| x.inspect}.join(', ')}"
end
def text(data)
return if data =~ /^\w*$/ # whitespace only
abbrev = data[0..40] + (data.length > 40 ? "..." : "")
puts " text : #{abbrev.inspect}"
end
end
list = MyListener.new
xmlfile = File.new("movies.xml")
Document.parse_stream(xmlfile, list)
```

以上输出结果为：

```
tag_start: "collection", {"shelf"=>"New Arrivals"}
tag_start: "movie", {"title"=>"Enemy Behind"}
```

```
  tag_start: "type", {}
    text   :   "War, Thriller"
  tag_start: "format", {}
  tag_start: "year", {}
  tag_start: "rating", {}
  tag_start: "stars", {}
  tag_start: "description", {}
    text   :   "Talk about a US-Japan war"
  tag_start: "movie", {"title"=>"Transformers"}
  tag_start: "type", {}
    text   :   "Anime, Science Fiction"
  tag_start: "format", {}
  tag_start: "year", {}
  tag_start: "rating", {}
  tag_start: "stars", {}
  tag_start: "description", {}
    text   :   "A schientific fiction"
  tag_start: "movie", {"title"=>"Trigun"}
  tag_start: "type", {}
    text   :   "Anime, Action"
  tag_start: "format", {}
  tag_start: "episodes", {}
  tag_start: "rating", {}
  tag_start: "stars", {}
  tag_start: "description", {}
    text   :   "Vash the Stampede!"
  tag_start: "movie", {"title"=>"Ishtar"}
  tag_start: "type", {}
  tag_start: "format", {}
  tag_start: "rating", {}
  tag_start: "stars", {}
  tag_start: "description", {}
    text   :   "Viewable boredom"
```

XPath 和 Ruby

我们可以使用XPath来查看XML ,XPath 是一门在 XML 文档中查找信息的语言(查看：XPath 教程)。

XPath即为XML路径语言，它是一种用来确定XML（标准通用标记语言的子集）文档中某部分位置的语言。XPath基于XML的树状结构，提供在数据结构树中找寻节点的能力。

Ruby 通过 REXML 的 XPath 类支持 XPath，它是基于树的分析（文档对象模型）。

实例

```ruby
#!/usr/bin/ruby -w
require 'rexml/document'
include REXML
xmlfile = File.new("movies.xml")
xmldoc = Document.new(xmlfile)
```

```
# 第一个电影的信息
movie = XPath.first(xmldoc, "//movie")
p movie
# 打印所有电影类型
XPath.each(xmldoc, "//type") { |e| puts e.text }
# 获取所有电影格式的类型，返回数组
names = XPath.match(xmldoc, "//format").map {|x| x.text }
p names
```

以上实例输出结果为：

```
<movie title='Enemy Behind'> ... </>
War, Thriller
Anime, Science Fiction
Anime, Action
Comedy
["DVD", "DVD", "DVD", "VHS"]
```

XSLT 和 Ruby

Ruby 中有两个 XSLT 解析器，以下给出简要描述：

Ruby-Sablotron

这个解析器是由正义Masayoshi Takahash编写和维护。这主要是为Linux操作系统编写的，需要以下库：

- Sablot
- Iconv
- Expat

你可以在 Ruby-Sablotron 找到这些库。

XSLT4R

XSLT4R 由 Michael Neumann 编写。 XSLT4R 用于简单的命令行交互，可以被第三方应用程序用来转换XML文档。
XSLT4R需要XMLScan操作，包含了 XSLT4R 归档，它是一个100%的Ruby的模块。这些模块可以使用标准的Ruby安装方法（即Ruby install.rb）进行安装。

XSLT4R 语法格式如下：

```
ruby xslt.rb stylesheet.xsl document.xml [arguments]
```

如果您想在应用程序中使用XSLT4R，您可以引入XSLT及输入你所需要的参数。实例如下：

实例

```
require "xslt"
stylesheet = File.readlines("stylesheet.xsl").to_s
xml_doc = File.readlines("document.xml").to_s
arguments = { 'image_dir' => '/....' }
sheet = XSLT::Stylesheet.new( stylesheet, arguments )
# output to StdOut
```

```
sheet.apply( xml_doc )
# output to 'str'
str = ""
sheet.output = [ str ]
sheet.apply( xml_doc )
```

更多资料

- 完整的 REXML 解析器, 请查看文档 REXML 解析器文档。
- 你可以从 RAA 知识库 中下载 XSLT4R 。

← Ruby Socket 编程 Ruby Web Service 应用 – SOAP4R →