文章
姚 鑫 · 五月 21, 2021 阅读大约需 6 分钟

第二章 设置和获取HTTP标头

第二章 设置和获取HTTP标头

设置和获取HTTP标头

可以设置和获取HTTP标头的值。

%Net.HttpRequest的以下每个属性都包含具有相应名称的HTTP标头的值。如果不设置这些属性，则会自动计算它们：

- Authorization
- ContentEncoding
- ContentLength(此属性为只读。)
- ContentType (指定Content-Type标头的Internet媒体类型(MIME类型)。)
- ContentCharset (指定Content-Type标题的字符集部分。如果设置此属性，则必须首先设置ContentType属性。)
- Date
- From
- IfModifiedSince
- Pragma
- ProxyAuthorization
- Referer
- UserAgent

%Net.HttpRequest类提供可用于设置和获取主HTTP标头的常规方法。这些方法忽略Content-Type和其他实体标头。

ReturnHeaders()

返回包含此请求中的主HTTP标头的字符串。

OutputHeaders()

将主HTTP标头写入当前设备。

GetHeader()

返回此请求中设置的任何主HTTP标头的当前值。此方法接受一个参数，即头的名称(不区分大小写)；这是一个字符串，如Host或Date

SetHeader()

设置标题的值。通常，可以使用它来设置非标准标头；大多数常用标头都是通过Date等属性设置的。此方法有两个参数：

- 标头的名称(不区分大小写)，不带冒号(：)分隔符；这是一个字符串，如Host或Date
- 标头值

不能使用此方法设置实体标头或只读标头(Content-Length和Connection)。

管理保活（Keep-alive）行为

如果重复使用%Net.HttpRequest的同一实例来发送多个HTTP请求，则默认情况下，InterSystems IRIS会使TCP/IP套接字保持打开状态，这样InterSystems IRIS就不需要关闭并重新打开它。

如果不想重复使用TCP/IP套接字，请执行以下任一操作：

- 设置SocketTimeout属性为0。
- 在你的HTTP请求中添加'Connection: close' HTTP头。
  要做到这一点，在发送请求之前添加如下代码:

```
Set sc=http.SetHeader("Connection","close")
```

注意，每个请求之后都会清除HTTP请求头，因此需要在每个请求之前包含此代码。

%Net.HttpRequest的SocketTimeout属性指定InterSystems IRIS将重用给定套接字的时间窗口(以秒为单位)。此超时旨在避免使用可能已被防火墙静默关闭的套接字。此属性的默认值为115。可以将其设置为不同的值。

处理HTTP请求参数

发送HTTP请求时(请参阅 “ 发送HTTP请求 ”)，可以在位置参数中包括参数；例如："/test.html?PARAM=%25VALUE" 将PARAM设置为等于%value。

还可以使用以下方法控制%Net.HttpRequest实例处理参数的方式：

InsertParam()

将参数插入到请求中。此方法接受两个字符串参数：参数的名称和参数的值。例如：

```
 do req.InsertParam("arg1","1")
```

可以为给定参数插入多个值。如果这样做，这些值将接收从1开始的下标。在其他方法中，可以使用这些下标来引用目标值。

DeleteParam()

从请求中删除参数。第一个参数是参数的名称。第二个参数是要删除的值的下标；仅当请求包含同一参数的多个值时才使用此参数。

CountParam()

统计与给定参数关联的值数。

GetParam()

获取请求中给定参数的值。第一个参数是参数的名称。如果请求没有同名的参数，则第二个参数是要返回的默认值；该默认值的初始值为空值。第三个参数是要获取的值的下标；仅当请求包含同一参数的多个值时才使用此参数。

IsParamDefined()

检查是否定义了给定参数。如果参数有值，则此方法返回TRUE。参数与DeleteParam()相同。

NextParam()

通过$order()对参数名称进行排序后，检索下一个参数的名称(如果有)。

ReturnParams()

返回此请求中的参数列表。

包括请求正文

HTTP请求可以包括请求正文或表单数据。要包括请求正文，请执行以下操作：

1. 创建%GlobalBinaryStream的实例或子类。将此实例用于HTTP请求的EntityBody属性。
2. 使用标准流接口将数据写入此流。例如：

```
Do oref.EntityBody.Write("Data into stream")
```

例如，可以读取一个文件并将其用作自定义HTTP请求的实体正文：

```
set file=##class(%File).%New("G:\customer\catalog.xml")
set status=file.Open("RS")
if $$$ISERR(status) do $System.Status.DisplayError(status)
set hr=##class(%Net.HttpRequest).%New()
do hr.EntityBody.CopyFrom(file)
do file.Close()
```

发送分块请求

如果使用的是HTTP1.1，则可以分块发送HTTP请求。这涉及到设置Transfer-Encoding以指示消息已分块，并使用大小为零的块来指示完成。

当服务器返回大量数据并且在完全处理请求之前不知道响应的总大小时，分块编码非常有用。在这种情况下，通常需要缓冲整个消息，直到可以计算出内容长度(%Net.HttpRequest会自动计算)。

要发送分块请求，请执行以下操作：

1. 创建%Net.ChunkedWriter的子类，%Net.ChunkedWriter是定义以块形式写入数据的接口的抽象流类。在这个子类中，实现OutputStream()方法。
2. 在%Net.HttpRequest的实例中，创建%Net.ChunkedWriter子类的实例，并用要发送的请求数据填充它。
3. 将%Net.HttpRequest实例的EntityBody属性设置为等于此%Net.ChunkedWriter实例。

当发送HTTP请求时(请参见“ 发送HTTP请求 ”)，它将调用EntityBody属性的OutputStream()方法。

在%Net.ChunkedWriter的子类中，OutputStream()方法应该检查流数据，决定是否分块以及如何分块，并调用类的继承方法来编写输出。

有以下方法可用：

WriteSingleChunk()

接受字符串参数并将该字符串作为非分块输出写入。

WriteFirstChunk()

接受字符串参数。写入适当的Transfer-Encoding标题以指示分块的消息，然后将字符串作为第一个分块写入。

WriteChunk()

接受字符串参数并将字符串作为块写入。

WriteLastChunk()

接受字符串参数，并将字符串作为块写入，后跟零长度块以标记结尾。

如果非NULL，则TranslateTable属性指定用于在写入时转换每个字符串的转换表。前面的所有方法都检查此属性。

发送表单数据

HTTP请求可以包括请求正文或表单数据。要包括表单数据，请使用以下方法：

InsertFormData()

将表单数据插入到请求中。此方法接受两个字符串参数：表单项的名称和关联值。可以为给定表单项插入多个值。如果这样做，值将接收从1开始的下标。在其他方法中，可以使用这些下标来引用目标值

DeleteFormData()

从请求中删除表单数据。第一个参数是表单项的名称。第二个参数是要删除的值的下标；仅当请求包含同一表单项的多个值时才使用此参数。

CountFormData()

统计请求中与给定名称关联的值数。

IsFormDataDefined()

检查是否定义了给定的名称

NextFormData()

通过$order()对名称进行排序后，检索下一个表单项的名称(如果有)。

例1

插入表单数据后，通常调用Post()方法。例如：

```
Do httprequest.InsertFormData("element","value")
Do httprequest.Post("/cgi-bin/script.CGI")
```

例2

```
Set httprequest=##class(%Net.HttpRequest).%New()
set httprequest.SSLConfiguration="MySSLConfiguration"
set httprequest.Https=1
set httprequest.Server="myserver.com"
set httprequest.Port=443
Do httprequest.InsertFormData("portalid","2000000")
set tSc = httprequest.Post("/url-path/")
Quit httprequest.HttpResponse
```

插入、列出和删除Cookie

%Net.HttpRequest自动管理从服务器发送的Cookie；如果服务器发送Cookie，%Net.HttpRequest实例将在下一次请求时返回此Cookie。(要使此机制正常工作需要重用%Net.HttpRequest的同一实例。)

使用以下方法管理%Net.HttpRequest实例中的Cookie：

InsertCookie()

将Cookie插入到请求中。指定以下参数：

- Cookie的名称。
- Cookie的值。
- 应存储Cookie的路径。
- 要从中下载Cookie的计算机的名称。
- Cookie过期的日期和时间。

GetFullCookieList()

返回Cookie的数量，并(通过引用)返回Cookie数组。

DeleteCookie()

请记住，Cookie是特定于HTTP服务器的。当插入Cookie时，使用的是到特定服务器的连接，而该Cookie在其他服务器上不可用。

#.NET #Caché #InterSystems IRIS #InterSystems IRIS for Health

---

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%BA%8C%E7%AB%A0-%E8%AE%BE%E7%BD%AE%E5%92%8C%E8%8E%B7%E5%8F%96http%E6%A0%87%E5%A4%B4