文章
姚 鑫 · 五月 23, 2021 阅读大约需 6 分钟

第三章 发送HTTP请求

第三章 发送HTTP请求

发送HTTP请求

创建HTTP请求后，使用以下方法之一发送该请求：

Delete()

```
method Delete(location As %String = "",
              test As %Integer = 0,
              reset As %Boolean = 1) as %Status
```

发出HTTP DELETE请求。

Get()

```
method Get(location As %String = "",
           test As %Integer = 0,
           reset As %Boolean = 1) as %Status
```

发出HTTP GET请求。此方法使Web服务器返回请求的页面。

Head()

```
method Head(location As %String,
            test As %Integer = 0,
            reset As %Boolean = 1) as %Status
```

发出HTTP Head请求。此方法使Web服务器仅返回响应头，而不返回正文。

Patch()

```
method Patch(location As %String = "",
             test As %Integer = 0,
             reset As %Boolean = 1) as %Status
```

发出HTTP修补程序请求。使用此方法可以对现有资源进行部分更改。

Post()

```
method Post(location As %String = "",
            test As %Integer = 0,
            reset As %Boolean = 1) as %Status
```

发出HTTP POST请求。使用此方法可将数据(如表单结果)发送到Web服务器，或上载文件。有关示例，请参阅 “ 发送表单数据 ” 。

Put()

```
method Put(location As %String = "",
           test As %Integer = 0,
           reset As %Boolean = 1) as %Status
```

发出HTTP PUT请求。使用此方法将数据上载到Web服务器。PUT请求并不常见。

Send()

```
method Send(type As %String,
            location As %String,
            test As %Integer = 0,
            reset As %Boolean = 1) as %Status
```

将指定类型的HTTP请求发送到服务器。此方法通常由其他方法调用，但如果要使用不同的HTTP谓词，则提供此方法以供使用。此处type是指定HTTP谓词(如 “ POST ”)的字符串。

在所有情况下：

- 每个方法都返回一个状态，应该检查该状态。
- 如果该方法正确完成，则对此请求的响应将位于HttpResponse属性中。
- Location参数是要请求的URL，例如："/test.html"。
- Location参数可以包含参数，假定这些参数已经URL转义，例如："/test.html?PARAM=%25VALUE"将PARAM设置为等于%VALUE。
- 使用test参数检查正在发送的是您预期要发送的内容：
    - 如果test为1，则该方法不会连接到远程计算机，而是将其本应发送到Web服务器的内容输出到当前设备。
    - 如果test为2，则在发出HTTP请求后将响应输出到当前设备。
- 在从服务器读取响应后，每个方法都会自动调用Reset()方法，除非test=1或Reset=0。

Reset()方法重置%Net.HttpRequest实例，以便它可以发出另一个请求。这比关闭此对象并创建新实例要快得多。这还会将Location标头的值移动到Referer标头。

```
Set httprequest=##class(%Net.HttpRequest).%New()
Set httprequest.Server="www.intersystems.com"
Do httprequest.Get("/")
```

创建和发送多部分POST请求

要创建和发送多部分POST请求，请使用%Net.MIMEPart类，本书后面将详细讨论这些类。下面的示例发送包含两个部分的POST请求。第一部分包括文件二进制数据，第二部分包括文件名。

```
ClassMethod CorrectWriteMIMEMessage3(header As %String)
{
     // Create root MIMEPart
     Set RootMIMEPart=##class(%Net.MIMEPart).%New()

     //Create binary subpart and insert file data
     Set BinaryMIMEPart=##class(%Net.MIMEPart).%New()
     Set contentdisp="form-data; name="_$CHAR(34)_"file"_$CHAR(34)_"; filename="
                    _$CHAR(34)_"task4059.txt"_$CHAR(34)
     Do BinaryMIMEPart.SetHeader("Content-Disposition",contentdisp)

     Set stream=##class(%FileBinaryStream).%New()
     Set stream.Filename="/home/tabaiba/prueba.txt"
     Do stream.LinkToFile("/home/tabaiba/prueba.txt")

     Set BinaryMIMEPart.Body=stream
     Do BinaryMIMEPart.SetHeader("Content-Type","text/plain")

    // Create text subpart
    Set TextMIMEPart=##class(%Net.MIMEPart).%New()
    Set TextMIMEPart.Body=##class(%GlobalCharacterStream).%New()
    Do TextMIMEPart.Body.Write("/home/tabaiba/prueba.txt")

    // specify some headers
    Set TextMIMEPart.ContentType="text/plain"
    Set TextMIMEPart.ContentCharset="us-ascii"
    Do TextMIMEPart.SetHeader("Custom-header",header)

    // Insert both subparts into the root part
    Do RootMIMEPart.Parts.Insert(BinaryMIMEPart)

    // create MIME writer; write root MIME message
    Set writer=##class(%Net.MIMEWriter).%New()

    // Prepare outputting to the HttpRequestStream
    Set SentHttpRequest=##class(%Net.HttpRequest).%New()
    Set status=writer.OutputToStream(SentHttpRequest.EntityBody)
    if $$$ISERR(status) {do $SYSTEM.Status.DisplayError(status) Quit}

    // Now write down the content
    Set status=writer.WriteMIMEBody(RootMIMEPart)
    if $$$ISERR(status) {do $SYSTEM.Status.DisplayError(status) Quit}

    Set SentHttpRequest.Server="congrio"
    Set SentHttpRequest.Port = 8080

    Set ContentType= "multipart/form-data; boundary="_RootMIMEPart.Boundary
    Set SentHttpRequest.ContentType=ContentType

    set url="alfresco/service/sample/upload.json?"
            _"alf_ticket=TICKET_caee62bf36f0ea5bd51194fce161f99092b75f62"

    set status=SentHttpRequest.Post(url,0)
    if $$$ISERR(status) {do $SYSTEM.Status.DisplayError(status) Quit}
}
```

访问HTTP响应

发送HTTP请求后，请求的HttpResponse属性将更新。此属性是%Net.HttpResponse的实例。本节介绍如何使用Response对象。它包括以下主题：

访问响应的数据

HTTP响应的正文包含在响应的Data属性中。此属性包含流对象(特别是%GlobalBinaryStream)。要使用此流，请使用标准流方法：Write()、WriteLine()、Read()、ReadLine()、Rewind()、MoveToEnd()和Clear()。还可以使用流的Size属性。

请求的ReadRawMode属性控制如何读取响应正文。

- 默认情况下，此属性为False，并且InterSystems IRIS假定正文在响应的HTTP标头中指定的字符集内(并相应地转换该字符集)。
- 如果此属性为true，InterSystems IRIS将以原始模式读取正文(不执行字符集转换)。

还可以使用OutputToDevice()方法，该方法将完整响应写入当前设备。标头的顺序与Web服务器生成的顺序不同。

下面是一个简单的示例，在该示例中，我们将响应流复制到文件并保存：

```
/// w ##class(PHA.TEST.HTTP).Stream()
ClassMethod Stream()
{
    set request=##class(%Net.HttpRequest).%New()
    set request.Server="tools.ietf.org"
    set request.Https=1
    set request.SSLConfiguration="yx"
    set status=request.Get("/html/rfc7158")
    if $$$ISERR(status) {
         do $system.OBJ.DisplayError()
    } else {
         set response=request.HttpResponse
    }

    Set file=##class(%FileCharacterStream).%New()
    set file.Filename="e:/temp/rfc7158.txt"
    set status=file.CopyFrom(response.Data)
    if $$$ISERR(status) {
         do $system.OBJ.DisplayError()
    }
    do file.%Close()
    q ""
}
```

按名称获取HTTP标头

%Net.HttpResponse类将其HTTP标头存储在InterSystems IRIS多维数组中。要访问标头，请使用以下方法：

GetHeader()

返回给定头的值。

GetNextHeader()

返回给定标头之后的下一个标头的名称。

这些方法中的每一个都只有一个参数，即HTTP标头的名称字符串。

还可以使用OutputHeaders()方法，该方法将HTTP标头写入当前设备(尽管它们的生成顺序不同)。

访问有关响应的其他信息

%Net.HttpResponse 类提供了存储HTTP响应其他特定部分的属性:
- StatusLine存储HTTP状态行，这是响应的第一行。
- StatusCode存储HTTP状态码。
- ReasonPhrase存储与StatusCode对应的人类可读的原因。
- ContentInfo存储关于响应体的附加信息。
- ContentType存储了Content-Type:标头的值。
- HttpVersion表示发送响应的web服务器所支持的HTTP版本。

#SQL #Caché #InterSystems IRIS #InterSystems IRIS for Health

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%B8%89%E7%AB%A0-%E5%8F%91%E9%80%81http%E8%AF%B7%E6%B1%82