記事
Shintaro Kaminaka · 2020年7月3日 17m read

InterSystems IRIS Open Authorization Framework（OAuth 2.0）の実装 - パート1

この記事と後続の2つの連載記事は、InterSystems製品ベースのアプリケーションでOAuth 2.0フレームワーク（簡略化のためにOAUTHとも呼ばれます）を使用する必要のある開発者またはシステム管理者向けのユーザーガイドを対象としています。

作成者：Daniel Kutac（InterSystemsシニアセールスエンジニア）

公開後の修正および変更の履歴

- 2016年8月3日 - 新しいバージョンのページを反映するため、Googleのクライアント設定のスクリーンショットを修正し、Google APIのスクリーンショットを更新しました。
- 2016年8月28日
  - Cache 2016.2でのJSON対応への変更を反映するため、JSON関連コードを変更しました。
- 2017年5月3日
  - Cache 2017.1でリリースされた新しいUIと機能を反映するため、テキストと画面を更新しました。
- 2018年2月19日 - 最新の開発内容を反映するために、CachéをInterSystems IRISに変更しました。 製品名は変更されていますが、この記事はすべてのInterSystems製品（InterSystems IRIS Data Platform、Ensemble、Caché）を対象としています。

パート1. クライアント

概要

これは、3部構成のInterSystemsによるOpen Authorization Frameworkの実装に関する連載記事の最初の記事です。

この最初のパートでは、このトピックについて簡単に紹介し、InterSystems IRISアプリケーションが認証サーバーのクライアントとして機能し、保護されたリソースを要求する簡単なシナリオを示します。

パート2ではより複雑なシナリオについて説明します。そこではInterSystems IRIS自体が認証サーバーとして機能するほか、OpenID Connectを介した認証サーバーとしても機能します。

このシリーズの最後のパートでは、OAUTHフレームワーククラスの個々の部分について説明します。それらはInterSystems IRISにより実装されているからです。

Open Authorization Framework[1]とは

皆さんの多くはすでにOpen Authorization Frameworkとその使用目的について聞いたことがあるかと思います。そのため、この記事では初めて同フレームワークを耳にした方のために簡単な要約を掲載します。

現在はバージョン2.0であるOpen Authorization Framework（OAUTH）は、クライアント（データを要求するアプリケーション）とリソース所有者（要求されたデータを保持するアプリケーション）の間に間接的な信頼を確立す

ることにより、主にWebベースのアプリケーションが安全な方法で情報を交換できるようにするプロトコルです。 この信頼自体は、クライアントとリソースサーバーの両方が認識して信頼する機関によって提供されます。 この機関は認証サーバーと呼ばれます。

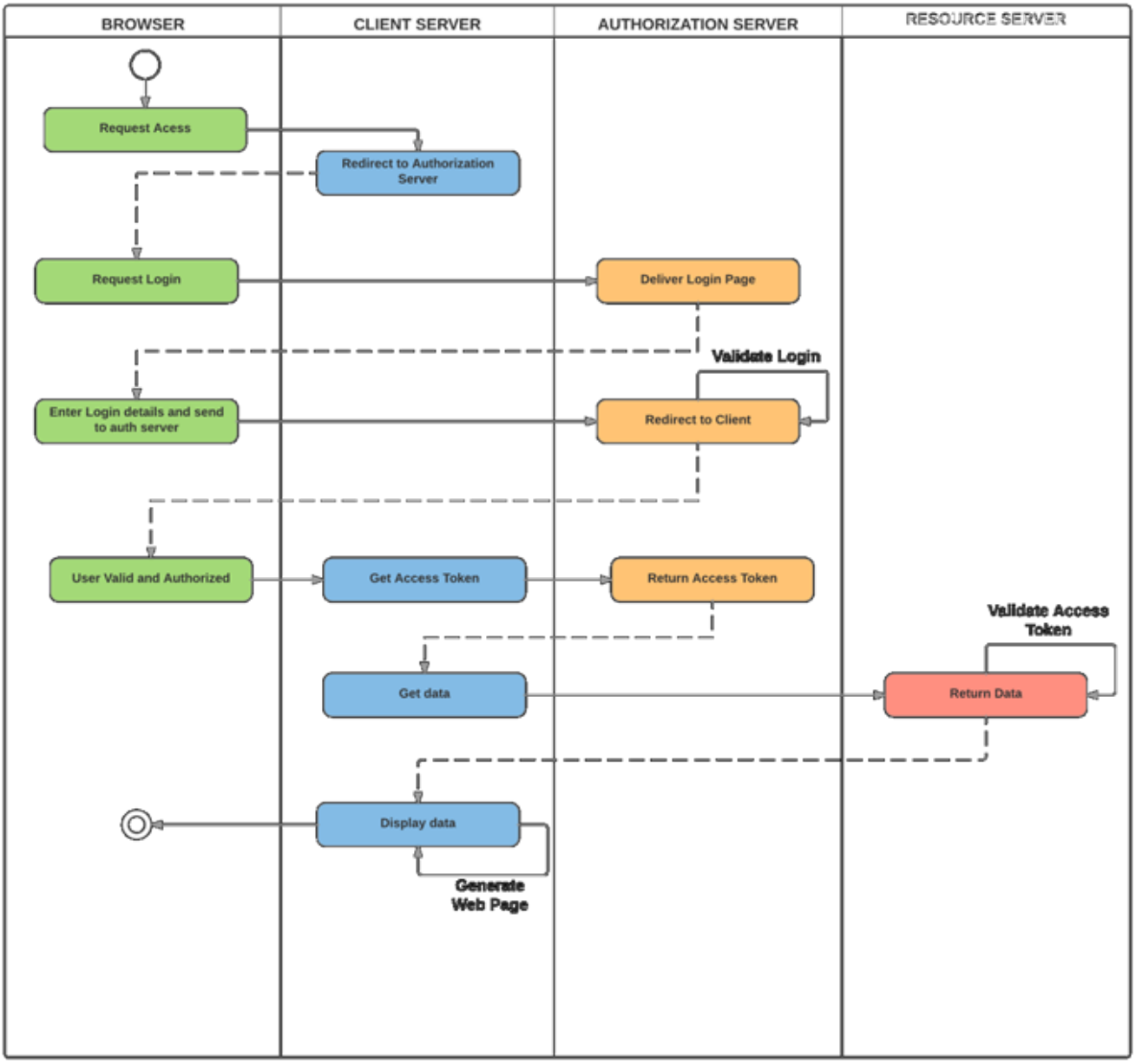
次の事例を使用して簡単に説明します。

Jenny（OAUTH用語ではリソース所有者）がJennyCorp社のプロジェクトに取り組んでいるとします。 彼女はより大きな潜在的なビジネスのプロジェクト計画を作成し、JohnInc社のビジネスパートナーであるJohn（クライアントユーザー）にドキュメントのレビューを依頼します。 ただし、彼女はジョンに自社のVPNへのアクセスを許可することを快く思っていないので、ドキュメントをGoogleドライブ（リソースサーバー）または他の同様のクラウドストレージに置いています。 そうすることで、彼女は彼女とGoogle（認証サーバー）の間に信頼関係を確立していました。 彼女はJohnと共有するドキュメントを選びます（JohnはすでにGoogleドライブを使用しており、Jennyは彼のメールアドレスを知っています）。

Johnはドキュメントを閲覧したいときには自分のGoogleアカウントで認証し、モバイルデバイス（タブレットやノートパソコンなど）からドキュメントエディタ（クライアントサーバー）を起動し、Jennyのプロジェクトファイルを読み込みます。

とてもシンプルに聞こえますが、2人とGoogleの間には多くの通信が発生しています。 どの通信もOAuth 2.0仕様に準拠しているため、Johnのクライアント（リーダーアプリケーション）は最初にGoogleで認証する必要があります（OAUTHはこのステップに対応していません）。ジョンがGoogleが提供するフォームにJohnが同意して認証すると、Googleはアクセストークンを発行し、リーダーアプリケーションにドキュメントへのアクセスを許可します。 リーダーアプリケーションはアクセストークンを使用してGoogleドライブにリクエストを発行し、Jennyのファイルを取得します。

以下の図に、個々の当事者間の通信を示しています。



注意: どのOAUTH 2.0通信もHTTPリクエストを使用していますが、サーバーは必ずしもWebアプリケーションである必要はありません。

InterSystems IRISを使ってこの簡単なシナリオを説明しましょう。

簡単なGoogleドライブのデモ

このデモでは、私たち自身のアカウントを使ってGoogleドライブに保存されているリソース（ファイルのリスト）をリクエストする小さなCSPベースのアプリケーションを作成します（ついでにカレンダーのリストも取得します）。

前提条件

アプリケーションのコーディングを始める前に、環境を準備する必要があります。
この環境には、SSLが有効になっているWebサーバーとGoogleのプロファイルが含まれます。

Webサーバーの構成

上記のように、認証サーバーとはSSLを使用して通信する必要があります。これは、OAuth 2.0がデフォルトでSSLを要求するためです。 データを安全に保つためには必要なことですよね？

この記事ではSSLをサポートするようにWebサーバーを構成する方法は説明しませんので、お好みの各Webサーバーのユーザーマニュアルを参照してください。 皆さんの好奇心をそそるため、この詳細な例ではMicrosoft IISサーバーを使用します（後でいくつかのスクリーンショットを掲載するかもしれません）。

Googleの構成

Googleに登録するには、Google API Manager（https://console.developers.google.com/apis/library?project=globalsummit2016demo）を使用する必要があります

デモのために、GlobalSummit2016Demoというアカウントを作成しました。 Drive APIが有効になっていることを確認してください。

次に、認証情報を定義します。

次の点に注意してください。

承認済みのJavaScript生成元 – 呼び出し元のページに対し、ローカルで作成されたスクリプトのみを許可します。

承認済みのリダイレクトURI – 理論上はクライアントアプリケーションを任意のサイトにリダイレクトできますが、InterSystems IRISのOAUTH実装を使用する場合は https://localhost/csp/sys/oauth2/OAuth2.Response.cls にリダイレクトする必要があります。 スクリーンショットのように複数の承認済みのリダイレクトURIを定義できますが、このデモでは2つのうち2番目のエントリのみが必要です。

最後に、InterSystems IRISをGoogle認証サーバーのクライアントとして構成する必要があります。

Cachéの構成

InterSystems IRIS
OAUTH2クライアントの構成は2段階で行われます。 まず、サーバー構成を作成する必要があります。

SMPで、System Administration（システム管理） > Security（セキュリティ） > OAuth 2.0 > Client Configurations（クライアント構成）を開きます。

「サーバー構成の作成」ボタンをクリックし、フォームに入力して保存します。

フォームに入力したすべての情報は、Google Developers Consoleのサイトで確認できます。 InterSystems IRISはOpen IDの自動検出に対応しています。 ただし、ここでは自動検出を使用せず、すべての情報を手動で入力しました。

次に、新しく作成された発行者エンドポイントの横にある「Client Configurations」（クライアント構成）リンクをクリックし、「Create Client Configuration」（クライアント構成を作成する）ボタンをクリックします。

「Client Information」（クライアント情報）タブと「JWT Settings」（JWT設定）タブは空のままにし（デフォルト値を使用します）、クライアントの認証情報を入力してください。

注意：ここでは、Confidential Clientを作成しています。これはPublic Clientよりも安全であり、クライアントシークレットがクライアントサーバーアプリケーションを離れることはありません（ブラウザに送信されません）。

また、「Use SSL/TLS」（SSL/TLSを使用する）がチェックされ、ホスト名（ここではクライアントアプリケーションにローカルにリダイレクトしているため、localhostにします）が入力され、さらにはポートとプレフィックスが入力されていることを確認してください（これは同じマシンに複数のInterSystems
IRISがある場合に役立ちます）。
入力した情報に基づいてクライアントリダイレクトURLが生成され、上の行に表示されます。

上のスクリーンショットでは、GOOGLEという名前のSSL構成を選択しました。 この名前自体は、多くのSSL構成のうちどれをこの特定の通信チャネルで使用するかを決定するためにのみ使用されます。
CachéはSSL/TLS構成を使用し、サーバー（この場合はGoogle OAuth 2.0
URI）との安全なトラフィックを確立するために必要なすべての情報を保存しています。

より詳細な説明については、ドキュメントを参照してください。

Googleの認証情報定義フォームから取得したクライアントIDとクライアントシークレットの値を入力します（手動構成を使用する場合）。

これですべての構成ステップが完了し、CSPアプリケーションのコーディングに進むことができます。

クライアントアプリケーション

クライアントアプリケーションは、シンプルなWebベースのCSPアプリケーションです。 そのため、Webサーバーによって定義および実行されるサーバー側のソースコードと、Webブラウザによってユーザーに公開されるユーザーインターフェイスで構成されています。

クライアントサーバー

クライアントサーバーは単純な2ページのアプリケーションです。
アプリケーション内では次の処理を実行します。

· Google認証サーバーのリダイレクトURLを組み立てます。

· Google Drive APIおよびGoogle Calendar APIへのリクエストを実行し、結果を表示します。

ページ1

これはアプリケーションの1ページであり、そのリソースについてGoogleを呼び出すことにしました。

以下はこのページを表す最小限の、しかし完全に動作するコードです。

```
Class Web.OAUTH2.Google1N Extends %CSP.Page

{


Parameter OAUTH2CLIENTREDIRECTURI = "https://localhost/csp/google/Web.OAUTH2.Google2N
.cls";


Parameter OAUTH2APPNAME = "Google";


ClassMethod OnPage() As %Status
```

```
{

  &html<<html>

<head>

</head>

<body style="text-align: center;">

        <!-- ?????????????? -->

        <h1>Google OAuth2 API</h1>

        <p>???????????OAuth2???????Google API???????????????

        <p>??????????Google???????????????????????????????

        >


  // Google???????openid???????????????

  set scope="openid https://www.googleapis.com/auth/userinfo.email "_

  "https://www.googleapis.com/auth/userinfo.profile "_

  "https://www.googleapis.com/auth/drive.metadata.readonly "_

  "https://www.googleapis.com/auth/calendar.readonly"


  set properties("approval_prompt")="force"

  set properties("include_granted_scopes")="true"


  set url=##class(%SYS.OAuth2.Authorization).GetAuthorizationCodeEndpoint(..#OAUTH2AP
PNAME,scope,

    ..#OAUTH2CLIENTREDIRECTURI,.properties,.isAuthorized,.sc)

  w !,"<p><a href='"_url_"'><img border='0' alt='Google?????' src='images/google-
signin-button.png' ></a>"


      &html<</body>

</html>>

  Quit $$$OK

}


ClassMethod OnPreHTTP() As %Boolean [ ServerOnly = 1 ]

{
```

```
  #dim %response as %CSP.Response

  set scope="openid https://www.googleapis.com/auth/userinfo.email "_

    "https://www.googleapis.com/auth/userinfo.profile "_

    "https://www.googleapis.com/auth/drive.metadata.readonly "_

    "https://www.googleapis.com/auth/calendar.readonly"

  if ##class(%SYS.OAuth2.AccessToken).IsAuthorized(..#OAUTH2APPNAME,,scope,.accessTok
en,.idtoken,.responseProperties,.error) {

    set %response.ServerSideRedirect="Web.OAUTH2.Google2N.cls"

  }

  quit 1

  }

}
```

以下にこのコードの簡単な説明を記します。

1. OnPreHTTPメソッド - まず、すでに有効なアクセストークンをGoogleの認証結果として取得しているかどうかを確認します。この認証は、例えば単にページを更新したときに発生する可能性があります。 トークンを取得できていない場合は、認証する必要があります。 トークンを取得できている場合は、結果表示ページにページをリダイレクトするだけです。

2. OnPageメソッド - 有効なアクセストークンがない場合にのみここに到達します。その場合、認証してGoogleに対する権限を付与し、アクセストークンを付与してもらうために通信を開始しなければなりません。

3. Google認証ダイアログの動作を変更するスコープ文字列とプロパティ配列を定義します（私たちのIDに基づいて認証する前に、Googleに対して認証する必要があります）。

4. 最後にGoogleのログインページのURLを受け取り、それをユーザーに提示してから同意ページを表示します。

追加の注意事項：

ここでは実際のリダイレクトページを https://www.localhost/csp/google/Web.OAUTH2.Google2N.cls （OAUTH2CLIENTREDIRECTURIパラメータ内）で指定しています。 ただし、Google認証情報の定義ではInterSystems IRIS OAUTH Frameworkのシステムページを使用しています。 リダイレクトは、OAUTHハンドラークラスによって内部的に処理されます。

ページ2

このページにはGoogle認証の結果が表示されます。成功した場合はGoogleのAPIコールを呼び出してデータを取得します。 繰り返しになりますが、このコードは最小限でも完全に機能します。
受信データがどのような構造で表示されるかは、皆さんのご想像にお任せします。

```
Include %occInclude


Class Web.OAUTH2.Google2N Extends %CSP.???
```

```
{


Parameter OAUTH2APPNAME = "Google";


Parameter OAUTH2ROOT = "https://www.googleapis.com";


ClassMethod OnPage() As %Status

{

  &html<<html>

   <head>

   </head>

   <body>>


  // ??????????????????

  set scope="openid https://www.googleapis.com/auth/userinfo.email "_

    "https://www.googleapis.com/auth/userinfo.profile "_

    "https://www.googleapis.com/auth/drive.metadata.readonly "_

    "https://www.googleapis.com/auth/calendar.readonly"


  set isAuthorized=##class(%SYS.OAuth2.AccessToken).IsAuthorized(..#OAUTH2APPNAME,,sc
ope,.accessToken,.idtoken,.responseProperties,.error)

  if isAuthorized {

    // Google????????????????????????????????????????????????????????????????RFC
 7662??????????

    w "<h3><span style='color:red;'>GetUserInfo API</span>??????</h3>"

    // userinfo?????API???????Get() ????????URL???????????????

    try {

    set tHttpRequest=##class(%Net.HttpRequest).%New()

      $$$THROWONERROR(sc,##class(%SYS.OAuth2.AccessToken).AddAccessToken(tHttpRequest
,"query","GOOGLE",..#OAUTH2APPNAME))

      $$$THROWONERROR(sc,##class(%SYS.OAuth2.AccessToken).GetUserinfo(..#OAUTH2APPNAM
E,accessToken,,.jsonObject))

      w jsonObject.%ToJSON()

    } catch (e) {
```

```
      w "<h3><span style='color: red;'>???: ",$zcvt(e.DisplayString(),"O","HTML")_"</
span></h3>"

    }

    /*****************************************

    *                                       *

    *      ??API?????????      *

    *                                       *

    *****************************************/

    w "<hr>"

    do ..RetrieveAPIInfo("/drive/v3/files")


    do ..RetrieveAPIInfo("/calendar/v3/users/me/calendarList")

  } else {

    w "<h1>??????????</h1>"

  }

  &html<</body>

  </html>>

  Quit $$$OK

}




ClassMethod RetrieveAPIInfo(api As %String)

{

  w "<h3><span style='color:red;'>"_api_"</span>??????</h3><p>"

  try {

    set tHttpRequest=##class(%Net.HttpRequest).%New()

    $$$THROWONERROR(sc,##class(%SYS.OAuth2.AccessToken).AddAccessToken(tHttpRequest,"
query","GOOGLE",..#OAUTH2APPNAME))

    $$$THROWONERROR(sc,tHttpRequest.Get(..#OAUTH2ROOT_api))
```

```
    set tHttpResponse=tHttpRequest.HttpResponse

    s tJSONString=tHttpResponse.Data.Read()

    if $e(tJSONString)'="{" {

      // JSON????

      d tHttpResponse.OutputToDevice()

    } else {

      w tJSONString

      w "<hr/>"

      /*

      // ???JSON API

      &html<<table border=1 style='border-collapse: collapse'>>

      s tJSONObject={}.%FromJSON(tJSONString)

      set iterator=tJSONObject.%GetIterator()

        while iterator.%GetNext(.key,.value) {

          if $isobject(value) {

            set iterator1=value.%GetIterator()

            w "<tr><td>",key,"</td><td><table border=1 style='border-
collapse: collapse'>"

            while iterator1.%GetNext(.key1,.value1) {

            if $isobject(value1) {

                set iterator2=value1.%GetIterator()

                w "<tr><td>",key1,"</td><td><table border=0 style='border-
collapse: collapse'>"

                while iterator2.%GetNext(.key2,.value2) {

                    write !, "<tr><td>",key2, "</td><td>",value2,"</td></tr>"


                }

                // ????????????????/?????????????

              w "</table></td></tr>"

            } else {

                  write !, "<tr><td>",key1, "</td><td>",value1,"</td></tr>"
```

```
            }

            }

          w "</table></td></tr>"

          } else {

              write !, "<tr><td>",key, "</td><td>",value,"</td></tr>"

          }

        }

    &html<</table><hr/>

    >

    */

    }

  } catch (e) {

    w "<h3><span style='color: red;'>???: ",$zcvt(e.DisplayString(),"O","HTML")_"</span></h3>"

  }

}


}
```

コードを簡単に見てみましょう。

1. まず、有効なアクセストークンがあるかどうかを確認する必要があります（そのため、認証を受けました）。

2. トークンがある場合はGoogleが提供し、発行されたアクセストークンがカバーするAPIにリクエストを発行できます。

3. そのためには標準の %Net.HttpRequest クラスを使用しますが、呼び出されたAPIの仕様に従ってGETメソッドまたはPOSTメソッドにアクセストークンを追加します。

4. ご覧のように、OAUTHフレームワークには GetUserInfo() メソッドが実装されていますが、RetrieveAPIInfo() ヘルパーメソッドの場合と同様に、Google APIの仕様を利用して直接ユーザー情報を取得できます。

5. OAUTHの世界ではJSON形式でデータを交換するのが一般的であるため、ここでは受信データを読み取り、それを単にブラウザに出力してます。 受信データを解析して整形し、それをユーザーが理解できる形で表示できるようにするのはアプリケーション開発者の責任です。 しかし、それはこのデモの範囲を超えています。
（ただし、いくつかのコメントアウトされたコードで構文解析のやり方を示しています。）

以下は、未加工のJSONデータが表示された出力のスクリーンショットです。

パート2に進んでください。そこでは、認証サーバーおよびOpenID Connectプロバイダーとして機能するInterSystems IRISについて説明します。

[1] https://tools.ietf.org/html/rfc6749、https://tools.ietf.org/html/rfc6750

#OAuth2 #セキュリティ #アクセス制御 #認証 #Caché #InterSystems IRIS

ソースURL:https://jp.community.intersystems.com/post/intersystems-iris-open-authorization-framework%EF%BC%88oauth-20%EF%BC%89%E3%81%AE%E5%AE%9F%E8%A3%85-%E3%83%91%E3%83%BC%E3%83%881