

Jakarta Commons HttpClient 3.x

(この文書は EMBL-EBI

http://www.ebi.ac.uk/Tools/webservices/tutorials/06_programming/java/rest/jakarta_commons_http_client_3.x

の和訳です。このドキュメントは、12/09/26 時点の情報にもとづいて書かれています。このドキュメントは ecobioinfo.com で独自に訳したもので、EMBL-EBI とは直接関係ありません。)

Java を使って HTTP (Hypertext Transfer Protocol) [1](#) [2](#) ベースのサービスへアクセスする一つの方法は Jakarta Commons HttpClient [4](#) を使うことです。REST Web サービスは HTTP ベースなので、Jakarta Commons HttpClient はどのような REST サービスのアクセスにも使うことができます。

インストール

Jakarta Commons HttpClient 3.x は、<http://hc.apache.org/httpclient-3.x/> からダウンロードできます。

Apache Ivy

[Apache Ivy](#) を使う場合は、下記の依存宣言が使えます:

```
<dependency org="commons-httpclient" name="commons-httpclient" rev="3.1" />
```

Apache Maven

[Apache Maven](#) を使う場合は、下記の依存宣言が使えます:

```
<dependency>
  <groupId>commons-httpclient</groupId>
  <artifactId>commons-httpclient</artifactId>
  <version>3.1</version>
  <scope>compile</scope>
</dependency>
```

HTTP GET

HTTP GET は最も単純な HTTP リクエストで、与えられた URL のドキュメントを獲得するために使われます。そのため GET を使うには、要求された Web サービスリソースの URL が必要とされます。サービスに依存しますが、おそらく静的な URL か、より一般的にはリクエストのパラメータに基づいて構築される URL でしょう。下記の用例は、[dbfetch](#), [WSDbfetch \(REST\)](#), [SRS](#) サービスを使った処理を説明しています。

dbfetch

dbfetch サービス (<http://www.ebi.ac.uk/Tools/dbfetch/dbfetch>) は、EMBL-EBI で利用可能な広範囲の生物学データベースから、識別子 (ID または Accession 番号) でデータエントリを獲得するための汎用的なインターフェースを提供しています。dbfetch にアクセスするためには、2 種類のスタ

イルの URL が使えます:

1. パラメータ化された URL:

```
http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?  
db={DB}&id={IDS}&format={FORMAT}&style={STYLE}
```

2. ドキュメントスタイル URL:

```
http://www.ebi.ac.uk/Tools/dbfetch/dbfetch/{DB}/{IDS}/{FORMAT}
```

dbfetch ドキュメント (<http://www.ebi.ac.uk/Tools/dbfetch/dbfetch>) には、データベース名 ({DB})、データフォーマット ({FORMAT})、データスタイル ({STYLE}) の有効値の詳細があります。識別子リスト ({IDS}) は、コマンドで区切られたエントリ識別子のリストです。識別子は、ID、名称、アクセスンNo が使えます。例えば、UniProtKB からラットとマウスの WAP タンパクを検索するには:

1. パラメータ化された URL: (訳注: 元ページで WAP_HUMAN になってますが WAP_MOUSE です)

```
http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?  
db=uniprotkb&id=WAP_RAT,WAP_HUMAN&format=uniprot&style=raw
```

2. ドキュメントスタイル URL:

```
http://www.ebi.ac.uk/Tools/dbfetch/dbfetch/uniprotkb/WAP_RAT,WAP_MOUSE/uniprot
```

dbfetch ドキュメントスタイル URL を使い UniProtKB WAP_RAT を取得する (<examples/rest/httpclient/DbfetchHttpClientGet.java>):

```
public class DbfetchHttpClientGet {  
    /** Get a web page using HTTP GET.  
     *  
     * @param urlStr The URL of the page to be retrieved.  
     * @return A string containing the page.  
     */  
    public static String getHttpUrl(String urlStr) {  
        // Data obtained from service, to be returned  
        String retVal = null;  
        // Create a client  
        HttpClient client = new HttpClient();  
        // Create a HTTP GET request  
        GetMethod method = new GetMethod(urlStr);  
        // Allow redirects to be followed  
        method.setFollowRedirects(true);  
        try {  
            // Execute the request using the client  
            int statusCode = client.executeMethod(method);  
            // Check the response status code  
            if (statusCode != HttpStatus.SC_OK) {  
                System.err.println("Method failed: " + method.getStatusLine());  
            }  
        }  
    }  
}
```

```

    }
    // Get the page data, allowing for character encoding
    BufferedReader bis = new BufferedReader(
        new InputStreamReader(
            method.getResponseBodyAsStream(),
            method.getResponseCharSet())
    );
    int bufLen = 8 * 1024;
    char[] charBuf = new char[ bufLen ];
    StringBuffer strBuf = new StringBuffer();
    int count;
    while( (count = bis.read(charBuf) ) != -1) {
        strBuf.append(charBuf, 0, count);
    }
    bis.close();
    retVal = strBuf.toString();
}
catch(HttpException ex) {
    System.out.println(ex.getMessage());
}
catch(IOException ex) {
    System.out.println(ex.getMessage());
}
finally {
    // Clean-up the connection
    method.releaseConnection();
}
// Return the response data
return retVal;
}

/** Execution entry point
 *
 * @param args Command-line arguments
 * @return Exit status
 */
public static void main(String[] args) {
    // Parameters for the dbfetch call
    String dbName = "uniprotkb"; // Database name (e.g. UniProtKB)
    String id = "WAP_RAT"; // Entry identifier, name or accession
    String format = "uniprot"; // Data format

    // Construct the dbfetch URL
    // Document style base URL
    String dbfetchBaseUrl = "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch/";
    // Add the database name, identifiers and format to the URL
    String dbfetchUrl = dbfetchBaseUrl + dbName + "/" + id + "/" + format;
}

```

```
// Get the page and print it.  
System.out.print(getHttpUrl(dbfetchUrl));  
}  
}
```

練習 1 : RESTful dbfetch

サンプルプロジェクトで dbfetch クライアントは ([examples/rest/httpclient/DbfetchHttpClientGet.java](#)) で提供されています。このクライアントからスタートして、[MBL-Bank](#) エントリの accession: M28668, M60493, M76128 を [dbfetch](#) を使って獲得してみましょう。

パラメータの適正な値とリクエスト URL の構造についての詳細は [dbfetch](#) と [WSDbfetch REST](#) のドキュメントを参照してください。

解答例 : [solutions/rest/httpclient/Q1DbfetchHttpClientGet.java](#)

SRS

[dbfetch](#) はエントリ獲得の有益なインターフェースを提供していますが、汎用的な検索要求システムではありません。検索要求処理の一選択肢として、SRS (<http://srs.ebi.ac.uk/>) があります。SRS は、一つのリクエストで複合的な複数データベースの検索要求を実行するのに役立つ URL ベースのインターフェースを提供しています。

SRS URL の最も簡単な書式は、DB:ID フォーマットでのエントリ識別子のリストの検索です。例えば “auxin” という単語を含んでいる [UniProtKB](#) のエントリの検索 :

```
http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?[uniprot-all:azurin*]
```

デフォルトでは先頭の 30 エントリしか返信されません。検索要求に一致したエントリの数を得るには “cResult” ページが使えます :

```
http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-page+cResult+[uniprot-all:azurin*]
```

これは検索要求に一致したエントリ数を返します :

```
170 entries for [uniprot-all:azurin*]
```

獲得された検索結果のエントリは、-bv を使用して先頭のエントリ番号を指定、-lv で獲得数を指定してまとめて取り出すことができます。最初から 30 エントリずつまとめて 2 回取り出すクエリの例 :

```
http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-bv+1+-lv+30+[uniprot-all:azurin*]  
http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-bv+31+-lv+30+[uniprot-all:azurin*]
```

検索要求にマッチしたエントリ識別子の獲得と同じように、完全なエントリは -e あるいはデータの特別な表示では -view を使って獲得することができます。例えば検索要求の結果の要約には

“SeqSimpleView”を使うことができます。

```
http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-view+SeqSimpleView+[uniprot-all:azurin*]
```

あるいは、fasta フォーマットのシーケンスを獲得するには“FastaSeqs”:

```
http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-view+FastaSeqs+[uniprot-all:azurin*]
```

SRS URL についての詳しい情報は [Linking to SRS guide](#) を参照してください。

(訳注: 本文では auxin と書いてありますが、例では azurin になっています。)

練習 2 : REST と SRS

サンプルプロジェクトで [dbfetch](#) クライアントは

([examples/rest/httpclient/DbfetchHttpClientGet.java](#)) で提供されています。このクライアントからスタートして、[EMBL Coding Sequences](#) データベース (EMBL CDS) から CFTR という名前の遺伝子を含むエントリの数を [SRS](#) を使って獲得してみましょう。

SRS の URL を構築する方法についての詳細は [Linking to SRS guide](#) クエリ文字列を構築する方法についての詳細は [SRS Query Language Quick Guide](#) を参照してください。

ヒント: SRS Web インターフェース (<http://srs.ebi.ac.uk/>) を使ってクエリを実行して、生成されたクエリ文字列を URL へコピーしましょう。

解答例 : [solutions/rest/httpclient/Q2SrsHttpClientGet.java](#)

HTTP POST

HTTP GET は情報検索のためには優れた手法ですが、GET を使用して送信できるデータ量には制限があります。そのため、大量のデータや複雑なパラメータの転送のために代替の方法を使う必要があります。HTTP POST は URL に無関係にデータを送信するので、POST は複雑なデータや大きいデータを転送する必要があるような状況で使われます。

dbfetch

[dbfetch](#) サービスが HTTP POST リクエストを HTTP GET リクエストと同様に受け入れる、これは識別子のリストを使うときに便利です。

例えば ([examples/rest/httpclient/DbfetchHttpClientPost.java](#))::

```
public class DbfetchHttpClientPost {  
    /**  
     * Get a web page using HTTP POST.  
     */  
}
```

```

* @param urlStr
*         The URL of the page to be retrieved.
* @param postData
*         Array of name value pairs describing the data for the POST
* @return A string containing the page.
*/
public static String getHttpUrlPost(String urlStr, NameValuePair[] postData)
{
    // Data obtained from service, to be returned
    String retVal = null;
    // Create a client
    HttpClient client = new HttpClient();
    // Create a HTTP POST request
    PostMethod method = new PostMethod(urlStr);
    // Add the POST data to the request
    method.setRequestBody(postData);
    try {
        // Execute the request using the client
        int statusCode = client.executeMethod(method);
        // Handle redirect response (cannot use setFollowRedirects(true)
        // with POST).
        // See http://hc.apache.org/httpclient-legacy/redirects.html
        if (statusCode == HttpStatus.SC_MOVED_PERMANENTLY
            || statusCode == HttpStatus.SC_MOVED_TEMPORARILY
            || statusCode == HttpStatus.SC_TEMPORARY_REDIRECT) {
            // Get the location to go to.
            Header locationHeader = method.getResponseHeader("location");
            if (locationHeader != null) {
                String redirectLocation = locationHeader.getValue();
                // Try request against new location
                method = new PostMethod(redirectLocation);
                method.setRequestBody(postData);
                statusCode = client.executeMethod(method);
            }
        }
        // Check the response status code
        if (statusCode != HttpStatus.SC_OK) {
            System.err.println("Method failed: " + method.getStatusLine());
        }
        // Get the page data, allowing for character encoding
        BufferedReader bis = new BufferedReader(new InputStreamReader(
            method.getResponseBodyAsStream(),
            method.getResponseCharSet()));
        int bufLen = 8 * 1024;
        char[] charBuf = new char[bufLen];
        StringBuffer strBuf = new StringBuffer();
        int count;
        while ((count = bis.read(charBuf)) != -1) {

```

```

        strBuf.append(charBuf, 0, count);
    }
    bis.close();
    retVal = strBuf.toString();
} catch (HttpException ex) {
    System.out.println(ex.getMessage());
} catch (IOException ex) {
    System.out.println(ex.getMessage());
} finally {
    // Clean-up the connection
    method.releaseConnection();
}
// Return the response data
return retVal;
}

/**
 * Execution entry point
 *
 * @param args
 *         Command-line arguments
 * @return Exit status
 */
public static void main(String[] args) {
    // Parameters for the dbfetch call
    String dbName = "uniprotkb"; // Database name (e.g. UniProtKB)
    String id = "WAP_RAT"; // Entry identifier, name or accession
    String format = "uniprot"; // Data format

    // Parameter style URL for dbfetch
    String dbfetchUrl = "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch";
    // Construct the POST data for the parameters
    NameValuePair[] postData = {
        new NameValuePair("db", dbName), // Database
        new NameValuePair("id", id), // Entry identifier(s)
        new NameValuePair("format", format), // Data format
        new NameValuePair("style", "raw") // Result style
    };

    // Get the page and print it.
    System.out.print(getHttpUrlPost(dbfetchUrl, postData));
}
}

```

プロキシ

一部の環境では、クライアントが外部のサービスに接続できるように HTTP プロキシを設定する必要があります。Jakarta commons HttpClient は HostConfiguration によるプロキシ設定をサポートしています:

```
// Create a client
HttpClient client = new HttpClient();
// Configure HTTP proxy.
HostConfiguration hostConf = client.getHostConfiguration();
hostConf.setProxy("proxy.example.org", 8080);
```

もし、Java システムプロパティ(例えば、`-Dhttp.proxyHost=proxy.example.org`
`-Dhttp.proxyPort=8080`) のサポートが必要とされるなら:

```
// Create a client
HttpClient client = new HttpClient();
// Configure HTTP proxy from system properties.
if(System.getProperty("http.proxyHost") != null) {
    String proxyHost = System.getProperty("http.proxyHost");
    int proxyPort =
        Integer.parseInt(System.getProperty("http.proxyPort"));
    HostConfiguration hostConf = client.getHostConfiguration();
    hostConf.setProxy(proxyHost, proxyPort);
}
```

User-Agent

HTTP クライアントは通常そのクライアントが何者かについての情報を提供し、もし必要なら特定のクライアントのみ異なる操作をするサービスを可能にし、サービスがどのように利用されているかについて、サービス提供者にいくらかの情報が与えられます。デフォルトで Java は HTTP User-Agent ヘッダ ([RFC2616 section 14.43](#) 参照) を Jakarta Commons-HttpClient/3.1、バージョン番号 (3.1) は HttpClient のバージョン、のように設定します。もし付加的にクライアントの識別が要求された場合、より詳述なプロダクトトークン ([RFC2616 section 3.8](#) 参照) が User-Agent 文字列の先頭に追加されなければなりません:

```
// Modify the user-agent to add a more specific prefix (see RFC2616 section 14.43)
HttpClientParams httpClientParams = new HttpClientParams();
httpClientParams.setParameter("http.useragent",
    "Example-Client/1.0 (" + System.getProperty("os.name") + ") "
    + httpClientParams.getParameter("http.useragent"));
HttpClient httpClient = new HttpClient(httpClientParams);
```

- 1) RFC1945 - Hypertext Transfer Protocol – HTTP/1.0 - <http://www.faqs.org/rfcs/rfc1945.html>
- 2) RFC2616 - Hypertext Transfer Protocol – HTTP/1.1 - <http://www.faqs.org/rfcs/rfc2616.html>
- 3) Java - <http://www.oracle.com/technetwork/java/index.html>
- 4) Jakarta Commons HttpClient 3.x - <http://hc.apache.org/httpclient-3.x/>