java.net

（この文書は EMBL-EBI http://www.ebi.ac.uk/Tools/webservices/tutorials/06_programming/java/rest/java.net の和訳です。このドキュメントは、12/09/24 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 [3] を使って HTTP（Hypertext Transfer Protocol）[1] [2] ベースのサービスにアクセスする手法の一つは、java.net パッケージで利用できる java.net.URL と java.net.HttpURLConnection クラスを使うことです。REST Web サービスは HTTP ベースなので、java.net.URL と java.net.HttpURLConnection 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

java.net パッケージは、Java ディストリビューションの一部として含まれています。個別のインストールは必要ありません。

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、コンマ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、アクセション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/javanet/RestDbfetchGet.java):

```
public class RestDbfetchGet {
    /** Get a web page using HTTP GET.
     *
     * @param urlStr The URL of the page to be retrieved as a string.
     * @return A string containing the page data.
     */
    public static String getHttpUrl(String urlStr) {
        // Data obtained from service, to be returned
        String retVal = null;
        // Get data using HTTP GET
        try {
            URL url = new URL(urlStr);
            BufferedReader inBuf = new BufferedReader(new
InputStreamReader(url.openStream()));
            StringBuffer strBuf = new StringBuffer();
            while(inBuf.ready()) {
                strBuf.append(inBuf.readLine() +
System.getProperty("line.separator"));
            }
            retVal = strBuf.toString();
        }
        catch(IOException ex) {
            System.out.println(ex.getMessage());
        }
        // 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 dbfetch call
        String dbName = "uniprot"; // Database name (e.g. UniProtKB)
        String id = "WAP_RAT"; // Entry identifier, name or accession
        String format = "uniprot"; // Data format

        // Construct the dbfetch URL
```

```
        // dbfetch 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/javanet/RestDbfetchGet.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/javanet/Q1RestDbfetchGet.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の数を返します：

```
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/javanet/RestDbfetchGet.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（EMBLCDS）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/javanet/Q2RestSrsGet.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わり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。

POST リクエストは、リクエストの”method”を明確に設定して POST データを与えなければならないのでもう少し複雑です（examples/rest/javanet/RestDbfetchPost.java）：

```
public class RestDbfetchPost {
/** Get a web page using HTTP POST.
 *
 * @param urlStr The URL of the page to be retrieved.
 * @param postStr String containing POST encoded data
 * @return A string containing the entry
 */
 public static String getHttpUrl(String urlStr, String postStr) {
  // Data obtained from service, to be returned
  String retVal = null;
  // Get data using HTTP POST
  try {
   // Create connection to URL
   URL url = new URL(urlStr);
   HttpURLConnection conn = (HttpURLConnection)url.openConnection();
   conn.setDoOutput(true);
   conn.setRequestMethod("POST");
   // Send POST data
   OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
   wr.write(postStr);
   wr.flush();
   // Get the response
   BufferedReader inBuf = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
   StringBuffer strBuf = new StringBuffer();
   while(inBuf.ready()) {
    strBuf.append(inBuf.readLine() + System.getProperty("line.separator"));
   }
   retVal = strBuf.toString();
  }
  catch(IOException ex) {
   System.out.println(ex.getMessage());
  }
  // Return the response data
  return retVal;
 }

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

```
  // Base URL for dbfetch
  String dbfetchUrlStr = "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch";
  // Construct POST data
  String dbfetchPostStr = null;
  try {
   // Ensure appropriate encoding is used.
   dbfetchPostStr = URLEncoder.encode("db", "UTF-8") + "=" +
URLEncoder.encode(dbName, "UTF-8")
    + "&" + URLEncoder.encode("id", "UTF-8") + "=" + URLEncoder.encode(id, "UTF-
8")
    + "&" + URLEncoder.encode("format", "UTF-8") + "=" +
URLEncoder.encode(format, "UTF-8")
    + "&" + URLEncoder.encode("style", "UTF-8") + "=" + URLEncoder.encode("raw",
"UTF-8");
  }
  catch(IOException ex) {
   System.out.println(ex.getMessage());
  }

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

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。Java はプロキシ通過の設定をサポートしています:

- Java システムプロパティ
  - JVM に設定

```
java -Dhttp.proxyHost=proxy.example.org -Dhttp.proxyPort=8080
ExampleClientClass
```

  - クライアントコードで設定

```
System.setProperty("http.proxyHost", "proxy.example.org");
System.setProperty("http.proxyPort", "8080");
```

- Java.net.Proxy と java.net.ProxySelector クラスを使用

詳細と実例は下記を参照:

- Networking Properties:
  http://download.oracle.com/javase/6/docs/technotes/guides/net/properties.html
- Java Networking and Proxies:
  http://download.oracle.com/javase/6/docs/technotes/guides/net/proxies.html

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 Java/1.6.0_13、バージョン番号（1.6.0_13）は Java 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)
String clientUserAgent = "Example-Client/1.0 (" + System.getProperty("os.name")
+ ")";
if (System.getProperty("http.agent") != null) {
    System.setProperty("http.agent", clientUserAgent + " " +
System.getProperty("http.agent"));
}
else {
    System.setProperty("http.agent", clientUserAgent);
}
```

注：この User-Agent 設定メソッドはグローバルで、コア Java パッケージを使って作られた全ての HTTP リクエストに影響します。しかし、サードパーティのパッケージでは User-Agent の設定に別の仕組みを使っていて変更が影響しないかもしれません。

---

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
