C/C++

Allgemeines

Der folgende C/C++-Quelltext demonstriert den Abruf von Daten (speziell des Views eines Datensatzes) über die API der Deutschen Digitalen Bibliothek. Zu Compilierung eines ausführbaren Kommandozeilenprogramms muss die cURL-Programmbibliothek vorhanden sein. Eine ausführbare Datei kann mit der freien GNU Compiler Collection (wie nachfolgend verwendet) als auch mit allen anderen C/C++-Compilern erstellt werden. Die Quelltextdatei `DDBRest.c` kann heruntergeladen und mit den folgenden Befehlen übersetzt sowie ausgeführt werden. Der Dateipfad des Compilers `gcc` sollte in der Umgebungsvariablen PATH des Betriebssystems definiert sein, andernfalls muss der vollständige Pfad mit Dateinamen (bspw. `/usr/bin/gcc`) angegeben werden.

Download

Da die Complierung des Programms etwas spezifischer als bei anderen Programmiersprachen ist, wird nun folgend der Weg unter Debian GNU/Linux bzw. dessen Derivate (z. B. Ubuntu) vorgestellt: Zunächst muss sichergestellt werden, dass die Programmbibliotheken von cURL der GNU Compiler Collection `gcc` zur Verfügung stehen. Da das Beispielprogramm Daten über eine gesicherte Verbindung (HTTP Secure) abruft, müssen zusätzlich die SSL-Bibliotheken vorhanden sein. Unter Debian stellt diese Abhängigkeiten das Software-Packet `libcurl4-openssl-dev` bereit. Dieses kann mit Hilfe der Paketverwaltung installiert werden. Bei der Erstellung eines lauffähigen Programms muss dem Linker mitgeteilt werden, dass die cURL-Bibliothek eingebunden werden muss. Dies erfolgt während des Compileraufrufs mit der Option `-l`. Die Benennung der ausführbaren Datei erfolgt mit der Option `-o`.

Debian GNU/Linux

```
> sudo apt-get install gcc libcurl4-openssl-dev
> gcc -l curl -o DDBRest DDBRest.c
> ./DDBRest
```

Es ist zu beachten, dass nur Daten abgefragt werden können, wenn ein gültiger API Key angegeben wird (Variable `key`). Andernfalls wird eine Fehlermeldung mit dem Hinweis „Response code was 403“ (Forbitten) ausgegeben.

Nach erfolgreichem Ausführen des Programms wird auf der Konsole dreimal der gleiche Datensatz ausgegeben:

1. Datensatz im XML-Format mit Authentifizierung über den HTTP request header
2. Datensatz im JSON-Format mit Authentifizierung über den HTTP request header
3. Datensatz im JSON-Format mit Authentifizierung über den Query Parameter

Dieses Beispiel ist selbstverständlich auch mit anderen Compilern (z. B. Visual C++), auch auf anderen Betriebssystemen in ein lauffähiges Programm übersetzbar. Wichtig ist, dass auch hier die cURL-Programmbibliotheken zur Verfügung stehen müssen. Weitere Hinweise zum Einbinden der Bibliotheken finden sich u. a. auf der Webseite von cURL.

Quelltext

DDBRest.c

```
#include <stdlib.h>
#include <string.h>
#include <curl/curl.h>

#define KEY "abcdefgh01234567"

struct url_data {
    size_t size;
    char* data;
};

static size_t write_data(char *buffer, size_t size, size_t nitems, void *userp)
```

```
{
    struct url_data *data = (struct url_data *)userp;
    size_t index = data->size;
    size_t n = (size * nitems);
    data->size += (size * nitems);
    char* tmp = realloc(data->data, data->size + 1);
    if(tmp) {
        data->data = tmp;
    } else {
        if(data->data) free(data->data);
        fprintf(stderr, "Failed to allocate memory.\n");
        return EXIT_FAILURE;
    }
    memcpy(data->data + index, buffer, n);
    data->data[data->size] = '\0';
    return size * nitems;
}

static size_t httpGet(const char* url, char* prop[], int prop_size, struct url_data
data)
{
    CURL *curl;
    CURLcode res;
    struct curl_slist* chunk = NULL;
    long http_code = 0;
    curl_global_init(CURL_GLOBAL_DEFAULT);
    curl = curl_easy_init();
    if(!curl) {
        fprintf(stderr, "cURL could not be initialized.");
        return EXIT_FAILURE;
    }
    curl_easy_setopt(curl, CURLOPT_URL, url);
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_data);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, &data);
    int i;
    for(i=0; i<prop_size; ++i) {
        chunk = curl_slist_append(chunk, prop[i]);
    }
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, chunk);
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, 0L);
    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYHOST, 0L);
    /* Perform the request, res will get the return code */
    res = curl_easy_perform(curl);
    curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &http_code);
    if(res != CURLE_OK) {
        fprintf(stderr, "%s\n", curl_easy_strerror(res));
        curl_easy_cleanup(curl);
        return EXIT_FAILURE;
    }
    if(http_code != 200) {
        fprintf(stderr, "Response code was %ld.\n", http_code);
        curl_easy_cleanup(curl);
        return EXIT_FAILURE;
    }
    curl_easy_cleanup(curl);
    curl_global_cleanup();
    return EXIT_SUCCESS;
}
```

```
int main(void)
{
    const char url[] =
"https://api.deutsche-digitale-bibliothek.de/items/OAXO2AGT7YH35YYHN3YKBXJMEI77W3FF/vi
ew";
    // get XML data via HTTP request header authentication
    struct url_data httpXmlData;
    httpXmlData.size = 0;
    httpXmlData.data = malloc(4096); /* reasonable size initial buffer */
    char* httpXmlResult[] = {"Authorization: OAuth oauth_consumer_key=\"" KEY "\"",
                             "Accept: application/xml"
                            };
    if(httpGet(url, httpXmlResult, sizeof(httpXmlResult)/sizeof(char*), httpXmlData)
== EXIT_SUCCESS) {
        fprintf(stdout, "%s\n", httpXmlData.data);
    }
    // get JSON data via HTTP request header authentication
    struct url_data httpJsonData;
    httpJsonData.size = 0;
    httpJsonData.data = malloc(4096); /* reasonable size initial buffer */
    char* httpJsonResult[] = {"Authorization: OAuth oauth_consumer_key=\"" KEY "\"",
                              "Accept: application/json"
                             };
    if(httpGet(url, httpJsonResult, sizeof(httpJsonResult)/sizeof(char*),
httpJsonData) == EXIT_SUCCESS) {
        fprintf(stdout, "%s\n", httpJsonData.data);
    }
    // get JSON data via query parameter authentication
    const char queryUrl[] =
"https://api.deutsche-digitale-bibliothek.de/items/OAXO2AGT7YH35YYHN3YKBXJMEI77W3FF/vi
ew?oauth_consumer_key=" KEY;
    struct url_data queryJsonData;
    queryJsonData.size = 0;
    queryJsonData.data = malloc(4096); /* reasonable size initial buffer */
    if(httpGet(queryUrl, NULL, 0, queryJsonData) == EXIT_SUCCESS) {
        fprintf(stdout, "%s\n", queryJsonData.data);
```

```
    }
    exit(EXIT_SUCCESS);
}
```