MySocials Libraries: Unified Access to Services of Social Networks

Sergey Zaharov, Anna Samoryadova, Pavel Shiryaev, Kirill Kulakov
Petrozavodsk State University
Petrozavodsk, Lenin st., 33, Russia
{szaharov, samoryad, shiryaev, kulakov}@cs.karelia.ru

Abstract

Internet services are one of the most important parts of platform ecosystem. There are several ways to access to Internet services: by system components, by standalone applications or by web browser. Most of them require a library (driver) to provide easy access to service functionality. In this paper we describe drivers for several social networks which are used to connect with each other, exchange opinions, photos and other media files. These drivers are a part of MySocials projects group. They use service Application Programming Interface (API) to access to service data and functionality. The driver's API based on XML structures. Each driver is created from a common template and has some changes corresponded service's API features. The current version of template supports various types of requests. Driver can upload and download files to service, open different web forms, send requests using GET and POST methods. The current version of MySocials project group contains drivers for four services: VKontakte, Facebook, Flickr and MyWorld from mail.ru.

Index Terms: social network, API, MySocials, driver

I. Introduction

Various applications and services are an important part of every software platform. They are responsible for user attraction. Many services can be found in Internet world. The main advantage of network service is collaboration with other users. For example, social networks give an opportunity to get information about your friends, exchange messages, share photos, videos, and listen to music on-line.

Network service provides a variety ways to work with it. The most of network services have web interface which is accessible through a browser. Another way is to create standalone application or system service which is connected to network service. In this case, user interface and some functions are separated from network service and it have to provide a light Application Programming Interface (API). Commonly it increases usability and reduces network traffic.

The main problem is that service has its own unique API. There are several differences between them. The first one is an authorization method. The service can use basic HTTP or digest access authentication [1], OAuth protocol [2], or some special analog of it. Authorization also could be passed with or without some type of encryption. The second difference is a request's structure. For example some services support complex requests. The third difference is a format of server's response. It could be plain text, XML-RPC, XML, JSON, etc.

All the services use structured text type for data transmission. It allows to process requests easily, modify them and transform them into some other kind. Therefore, it is possible to reduce service messages into unified form and then work with several different services by using some common protocol.

This paper describes implementation of methods used in MySocials project group [3] to access to social networks. We developed common API based on XML format for accessing services of social networks (MySocials API) and several libraries. MySocials project group contains libraries for VKontakte [4], Facebook [5], Flickr [6] and MyWorld from mail.ru [7].

The project was initiated at Petrozavodsk State University (PetrSU) FRUCT laboratory of wireless and mobile technologies [8]. MySocials belongs to the family of FRUCT research projects [9].

The rest of the paper is organized as follows. Section II briefly discusses the architecture of all MySocials libraries. Section III is dedicated to the description of special protocol for data transmission and the implementation of the drivers. Section IV summarizes our current results and future plans.

II. MySocials Driver Architecture

General MySocials driver interaction scheme is shown on Figure 1. Applications use MySocials API to get access to different social networks services. There are several drivers, libraries for interaction with social network services, based on this API. Each driver uses API of the service to access to data and functionality. All MySocials drivers are created from common template with some changes corresponded service's API features. Therefore, a new driver implemented using this template will be supported by all the applications based on MySocials API.

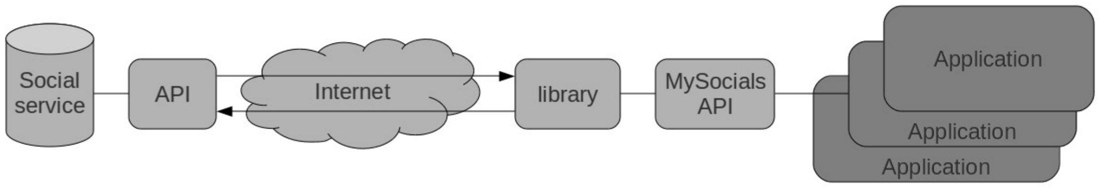

Fig. 1. General MySocials Driver interaction scheme

Each MySocials driver could be divided into two parts: common part, which includes base functionality, and special part, which is unique for specific service. Common part includes driver access functions, driver logic with common handlers, methods for working with accounts, authorization method and so on. Each handler can be overloaded. For example, service VKontakte sends text in response with unclassified HTML tag "<br>" and before next response processing handler has to close these tags. Therefore, there is no need to rewrite the base code to create a new driver.

MySocials drivers use special application WebAuth [10] to perform authorization requests. It is caused by the use of authorization through web interface by the most of social services. This application is able to represent any web page, in this use case it is an authorization page, and return the result which depends on definite conditions. Interaction with WebAuth is implemented by using D-Bus service. The driver passes URL of needed web page and some regular expression, which is used to define the end of authorization, to WebAuth and receives the result after that.

The architecture of MySocials driver is shown on Figure 2. The driver has a minimal set of interface functions, which provide the interaction with client applications: function for initialization of the driver, function for processing requests and function for shutdown the driver at the end of its work. During initialization driver creates new profile entity and adds it into a list of profiles. Profile structure contains account settings and other data needed for processing requests. In the request profile is identified by its unique ID. The profile entity is removed from the list at shutdown.

Request and response data presented as XML structure. It's flexible approach to modify and improve API without any modifications and recompilations for client applications. The another precondition is that most social services use XML structure to transfer data.

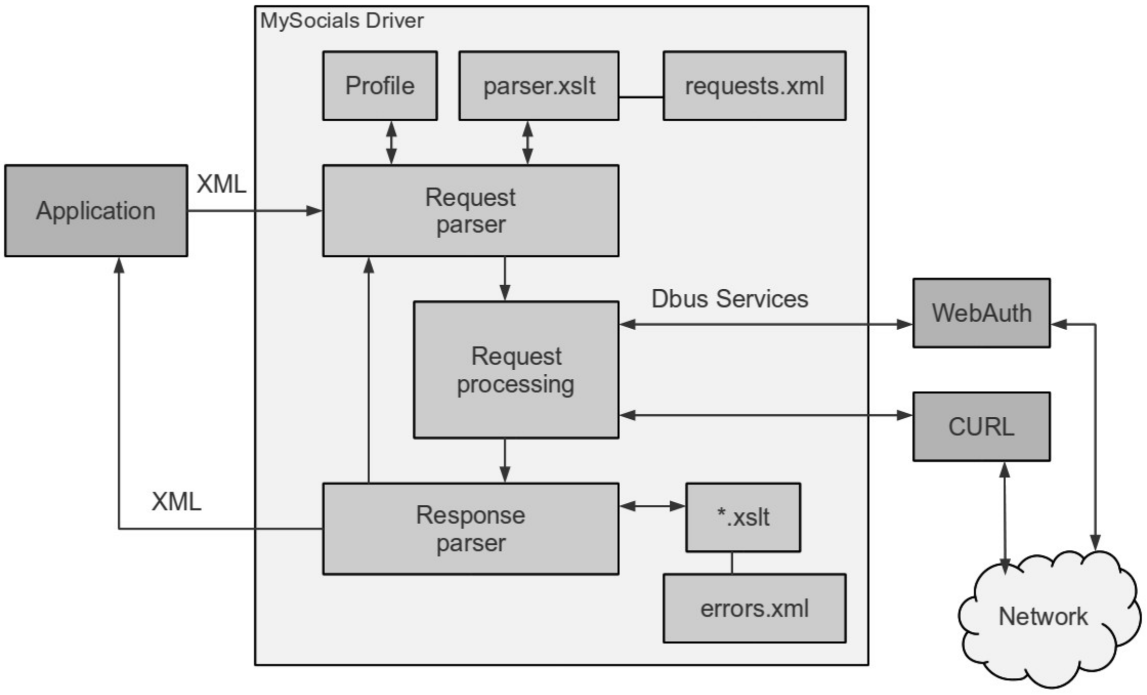

Fig. 2. Architecture of MySocials Driver

Request is transformed to response by using of XSLT transformations. MySocials drivers support processing of nested requests. It means that another request to the service could be constructed during parsing of service response using information from it. After that new request will be performed and the final response will be send to the client. Supported types of requests are described in Table I.

TABLE I
MySocials types of requests

Name of the type	Description
get	Simple GET request to the service.
post	POST request with some data included.
upload	POST request for uploading file to the service.
download	GET request for downloading file using direct link.
settings	Requests performed without sending data to the service.
webauth	Authorization request using WebAuth application.

If request needs a pretreatment then current transformation have to specify a corresponded handler. Handler modifies request content or takes some action. Each driver supports the following handlers:

- “settings” handler processes driver settings and updates profile data.
- “json” handler processes JSON response and converts it to XML.
- “empty” handler creates an empty response when no access to service is needed. The response fills on transformation.
- “download” handler downloads a file and stores it to stated path.
- “text” handler pretreatments request content (for example, closes “<br>” tags).
- “webauth” handler prepares request to WebAuth, parses response and stores profile if it is needed.

Common part contains default implementations of handlers enclosed preprocessor tags. When you need to re-implement handler you should to define corresponded macro definition and implement handler function in separate source file.

III. Implementation

During interaction with social services various data formats (XML, JSON, etc) and data structures can be used. Therefore, developers have to create special modules for working with different services, which provide processing of requests and responses. Developing of separate modules could lead to such difficulties as complexity of support and problems with introduction of new modules to a big program system. We developed common API for working with different social services to avoid all these difficulties. This API is based on XML data format, which allows us to transform information from various services to internal data view and backwards in very easy way. Therefore, MySocials API provides a good opportunity for developers to introduce drivers for interaction with different social services into their applications.

Request to MySocials driver has the following structure:

```
<Request class=" . . . " function = " . . . " noAuthorize="true"><Params> . . . </Params></Request>
```

Each request belongs to particular *class*, which includes a set of functions. Every *function* has its own set of parameters, they have to be described between *Params* tags. There is also an optional parameter *noAuthorize*. If this parameter’s value is true, it means that the driver isn’t able to call the application for authorization (WebAuth) and returns error message if authorization is requested.

Response from MySocials driver has the following structure:

```
<Response class=" . . . " function = " . . . " authorized="false"><Params> . . . </Params></Response>
```

The driver’s response is organized the same way as the request. Flag *authorized* has a true value when the driver performs authorization process successfully. This flag isn’t set if the error occurs during the request performing.

Description of all classes that MySocials drivers support is given in Table II.

TABLE II
Classes supported by MySocials drivers

Name of the class	Description
settings	Class for processing driver settings.
profile	Class for processing user profiles.
friends	Class for processing friends.
messages	Class for processing personal messages.
photos	Class for processing images and albums.
audio	Class for processing audio files.

We developed a special template of MySocials driver to make its implementation easier. This template includes a set of files with source code, which describe base logic of the driver’s work. This part can be changed taking consideration of some features of the interaction with definite service. The template also includes a set of XSLT files, which are needed to transform files into internal view. All these files can be modified easily to provide accordance with data structures, that service returns in its responses.

This template includes a special file “errors.xml”, which describes accordance between MySocials API errors and error messages using by social services.

Requests are described in “requests.xml” file and have the following structure:

```
<request class=" . . . " function=" . . . " access_type="public">
        <url sig="true"> . . . </url>
        <param multiple=" . . . " name=" . . . " select="."> . . . </param>
        <parser type=" . . . "> . . . </parser>
        <type> . . . </type>
</request>
```

Attribute *access_type* shows that this request can be called from client’s application. Tag *url* contains the URL for accessing to the service, attribute *sig* show if there is a need to include the signature into the request. Tag *param* contains request parameter and its default value. If the attribute *multiple* has true value, it means that current parameter is compound. Attribute *name* contains the name of parameter for request to the service and attribute *select* contains the name of parameter from client’s request. For example, next line describes service method for access to profile data.

```
<param name="method" select=".">flickr.people.getInfo</param>
```

Tag *parser* includes the name of the parser file for parsing service’s response, attribute *type* specifies the type of handler to use. Tag *type* contains the type of request (see Table I).

At the moment there are four drivers in MySocials project for interaction with such social services as VKontakte, Facebook, Flickr and MyWorld from mail.ru. Table III shows methods which are implemented in each driver.

Some methods are not supported by the drivers because some services don’t provide this functionality throw API. Nevertheless, these methods are included into MySocials API for future extensions. Another problem is in message processing. Some services use linear folders model to work with messages (for example, VKontakte and Flickr has inbox and outbox folders to store messages), but Facebook and MyWorld use threads which present messages as a tree. To solve this problem MySocials API contains both approaches to accessing messages.

Some methods haven't been fully implemented yet. For example, driver for MyWorld service doesn't completely support methods for obtaining lists of albums and images. We can only receive the user's own images. When such possibility will be added to the API of this service it will be implemented in the driver. We are planning to create a new class of methods for working with news and video.

TABLE III
MySocials drivers and supported methods

Class	Method	VKontakte driver	Facebook driver	Flickr driver	MyWorld driver
settings	Get driver settings	+	+	+	+
settings	Set driver settings	+	+	+	+
settings	Get list of supported methods	+	+	+	+
settings	Test connection to the service	+	+	+	+
profile	Get profile	+	+	+	+
profile	Get short version of profile	+	+	+	+
friends	Get a list of friends	+	+	+	+
friends	Delete a friend from list	-	-	-	-
photos	Get a list of albums	+	+	+	+/-
photos	Get a list of photos	+	+	+	+/-
photos	Get photo	+	+	+	+
photos	Uploading photos	+	+	-	+
photos	Create album	+	+	+	+
photos	Get list of photos which user marked	+	+	+	-
photos	Get lost of favorite photos	-	+	+	-
photos	Send comment to photos	+	+	+	-
messages	Get a list of private messages in folders	+	+	-	-
messages	Get list of private messages in threads	-	-	-	+
messages	Send private message	+	+	-	+
messages	Send a message to the wall	+	+	-	+
messages	Mark message as read	+	+	-	-
messages	Delete message	+	-	-	-
audio	Get a list of audio files	+	-	-	+

Driver core files contain 1931 lines of code. Implementation is made in the C language using glibc library. To access to service driver uses CURL. Proxy configuration is obtained from system environment and GConf. Data processing is available through libxml2, libxslt and JSON-C.

Code metrics of all drivers implementation are described in Table IV.

TABLE IV
MySocials drivers code metrics

Metric	VKontakte driver	Facebook driver	Flickr driver	MyWorld driver
LOC in parsers and configuration files	499	519	455	534
Number of XST files	41	30	19	23
Number of XML files	2	2	2	2
Number of supported methods	19	20	14	16

There are 13 tests with 47 examinations, which are implemented for automatic unit testing. MySocials driver console is a special application which is used for integration and interface testing of the drivers. The developer has to give the name of the driver and the name of the file with scenario for the driver to perform as parameters. All the responses from the driver can be written to a file. This approach allows to develop and test the driver separate of client's application.

The full code is available at Gitorious [11], a hosting for distributed open source projects. MySocials drivers are available for mobile platforms Maemo 5 and MeeGo and for desktop Linux based operation systems (e.g. OpenSUSE, Ubuntu, Fedora and Mandriva).

IV. Conclusion

There are many different services which provide similar features for communication, exchanging information and media-content. Protocols of interactions with services have a lot in common and their main difference is in data structures. Therefore, there is an opportunity to create unified API to work with different services. Our project implements such API and special template which allows to make process of realization of a driver much easier for the developers. This template contains interface functions, network functions, functions for working with user profiles, errors processing and methods for interaction with WebAuth. To create you own driver using the template all you need is to study service's API and implement parsers for reducing responses to MySocilas API format. Unified API provides an opportunity to include various drivers into one application in very simple way.

There are several applications use drivers to access to social services. There are sets of plugins for Qt Messaging Framework and Telepathy Framework; *MySocials* – a client of social network; *Sharing plugin for VKontakte service* – a plugin for standard image viewer on Maemo 5 platform and *MySocials Gallery* – an application for browsing images from different social services.

We are planning to improve the work of the existing drivers and implement new drivers which will provide the work with such popular social services as Picasa Web Albums [12], Photobuket [13], etc. There is also a need to improve some features of WebAuth client such as representation of title pages for each service and methods for working with CAPTCHA. MySocials API could be modified for its better work.

ACKNOWLEDGMENT

Authors would like to thank Finnish-Russian University Cooperation in Telecommunications (FRUCT) program for the provided support and R&D infrastructure. We would also like to thank Sergey Balandin, Timofey Turenko and Yurii Bogoyavlenskii for providing feedback and guidance.

REFERENCES

[1] Digest access authentication [Online]. Available: http://en.wikipedia.org/wiki/Digest_access_authentication
[2] Open Authorization [Online]. Available: http://en.wikipedia.org/wiki/OAuth
[3] MySocials development wiki. March. 2011. [Online]. Available: http://oss.fruct.org/wiki/MySocials/
[4] VKontakte. March. 2011. [Online]. Available: http://www.vkontakte.ru/
[5] Facebook. March. 2011. [Online]. Available: http://www.facebook.com/
[6] Flickr. March. 2011. [Online]. Available: http://www.flickr.com/
[7] MyWorld@mail.ru. March. 2011. [Online]. Available: http://my.mail.ru/
[8] Open Platforms for Mobile Devices. By Petrozavodsk State University, Department of Computer Science. March. 2011. [Online resource]. Available: http://oss.fruct.org/
[9] Open innovations framework program FRUCT, March. 2011. [Online]. Available: http://fruct.org/
[10] WebAuth [Online]. Available: http://oss.fruct.org/wiki/Webauth
[11] MySocials page at Gitorious. March. 2011. [Online]. Available: http://gitorious.org/mysocials/
[12] Picasa Web Albums. March. 2011. [Online]. Available: http://picasaweb.google.com/
[13] Image hosting, free photo sharing and video sharing at Photobucket. March. 2011. [Online]. Available: http://photobucket.com/