Insert here your thesis' task.

Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Bachelor's thesis

Adobe Premiere Pro Plugin for NARRA

Dmitry Vanyagin

Supervisor: Petr Pulc

12th April 2015

Acknowledgements

THANKS

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60(1) of the Act.

In Prague on 12th April 2015 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Vanyagin, Dmitry. *Adobe Premiere Pro Plugin for NARRA*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

Abstract

Summarize the contents and contribution of your work in a few sentences in English language.

Keywords Replace with comma-separated list of keywords in English.

Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova Replace with comma-separated list of keywords in Czech.

Contents

List of Figures

Introduction

The possibility to visualize and annotate data is always been appreciated in circles of people, who works with a huge amounts of information every day. NARRA is a software that provides this functionality for those using large amounts of video in their practice artists collaborating on open narratives, video editors, social scientists using video as a research tool, documentary filmmakers with expanded projects.

Artists can tell stories together using video. Instead of a linear narrative, media works can have multiple paths, multiple versions. The software can be used to create, visualize and navigate a tree of linked stories.

Video editors faced with hundreds of hours of material can annotate their media objects and then organize it based on complex search categories. The software itself will use existing software libraries to add functionality and perform automated annotations. For example the software can extract spoken words and make them into attached text; list shot size, geographic location, etc.

The main purpose of this thesis is to create a tool to use all this functionality and communicate with NARRA directly from your computer. I chose to implement it as a plug-in for Adobe Premiere Pro video editing software application. This application is widely used by different broadcasters such as the BBC[3] and CNN[5], also many people chose it as a part or their workflow. Using this plug-in user will be able to directly communicate with NARRA API, i.e., to import and export sequences of clips with all attached metadata.

To develop this tool I will use mainly Adobe Premiere Pro SDK and Third-party libraries for solving arising problems or for extending functionality provided by SDK.

Analysis

Requirements specification

Functional requirements

Final product have to fulfill basic requirements:

Authorization in NARRA User should be able to authorize in NARRA, get a token that will be used for all transactions between plug-in and NARRA API. Since Google identities are used inside NARRA, plug-in should communicate with Google authorization services.

Displaying a list of uploaded projects User should be able to browse a list of his projects that he exported to NARRA.

Searching a project by name Plug-in should provide convenient search input field for a user to search in a project list.

Importing a project from NARRA to Adobe Premiere Pro User should be able to choose a project from his project list and import it in the Adobe Premiere Pro.

Exporting a new project from Adobe Premiere Pro to NARRA User should be able to export a new project from Adobe Premiere Pro to NARRA.

Pushing changes of a project to NARRA User should be able to synchronize all changes made in project with NARRA.

Synchronization of project metadata between NARRA and local machine Plug-in should support synchronization of project metadata between NARRA and Adobe Premiere Pro project.

Non-Functional requirements

- User interface should be clear and understandable for a user.
- Plug-in should estimate the length operations and provide a visual representation (progress bars)
- Plug-in should work in Adobe Premiere Pro CS5.5 and newer versions.
- Internet connection is required to use all functionality.

Use cases

In this section I will try to describe the basic use cases for this application. In order to communicate with NARRA, user has to complete authorization process, that's why almost every use case require user to be logged in. You can see use case diagram on figure 0.1.

Authorization in NARRA User wants to authorize himself in order to use plug-in. Authorization process is carried out by Google.

Browsing project list User can see a list of his/her projects exported to NARRA. He/she is able to scroll the list, select a project. Requirements: User has to be logged in.

Import new project from NARRA User wants to import a project from NARRA. Project is selected from a list, after double click on the project, importing process starts, user sees progress bar of the import. Requirements: User has to be logged in.

Export project to NARRA User wants to export a project to NARRA. Current project in Adobe Premiere Pro will be pushed to NARRA after successful launch. User sees progress bar of export process with time estimation. Requirements: User has to be logged in.

Editing of metadata User wants to edit project metadata and save changes back to NARRA. User can add metadata using Adobe Premiere Pro tools. Requirements: User has to be logged in.

Searching for a project User wants to find a project in his project list. Name of the project should be written in a search box. List of projects will be automatically refreshed after each letter typed in search box. Requirements: User has to be logged in.

Structure of application

I decided to split this application into two plug-ins:

- Import plug-in (Importer).
- Export plug-in (Exporter).

The reason why I chose this structure is because it is more logical to have two lightweight plug-ins that solve it's own task. If user wants to import a project into Adobe Premiere Pro, he/she can launch Importer, to export a project Exporter can be used. On figure 0.2 you can see a sketch of communication between application and NARRA, you can see that plug-in directly uses Adobe Premiere Pro SDK and communicates with NARRA using http requests. All video files are stored in a cloud storage.

Authorization

To communicate with NARRA, user has to be authorized. NARRA uses google identities as a user objects, that is the reason why I have to somehow embed Google authorization service into this plug-in. I decided to do by using a web browser and login form provided by Google. Basic scenarion will be like this:

1. User launches plug-in.
2. Browser window opens up and request is sent to Google to start authorization process.
3. User sees Google login form.
4. User enters his/her username and password and allows plug-in to use profile data.
5. Plug-in receives token that will be used for communication with NARRA.

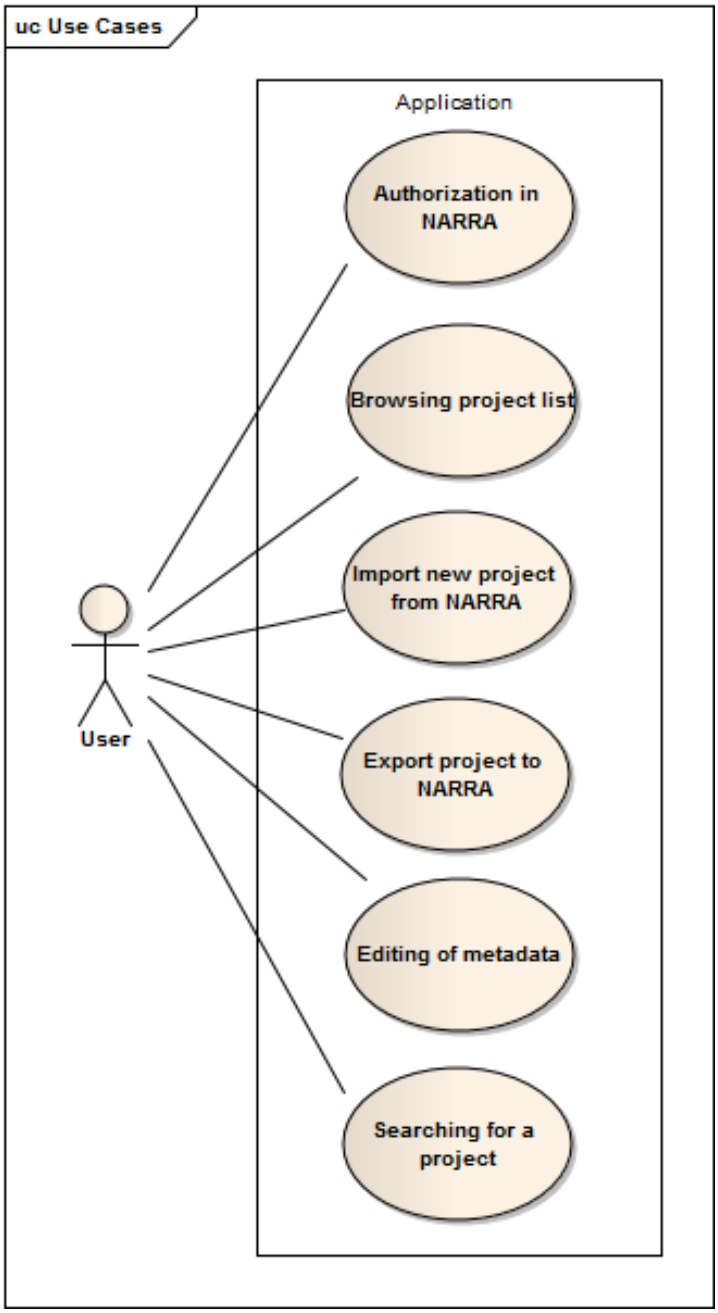

Figure 0.1: Use case diagram

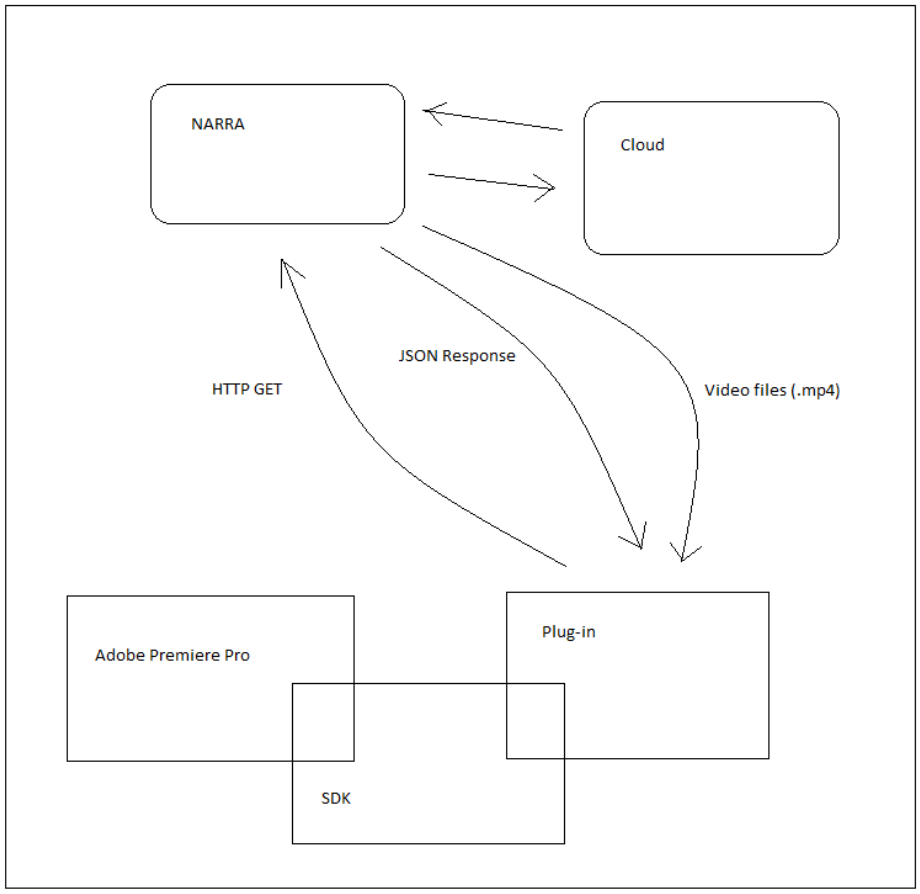

Figure 0.2: Sketch of communication with NARRA

OAuth 2.0

OAuth is an open standard for authorization. OAuth provides client applications a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials.

Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner, or end-user. The client then uses the access token to access the protected resources hosted by the resource server. OAuth is commonly used as a way for web surfers to log into third party web sites using their Microsoft, Google, Facebook or Twitter accounts, without worrying about their access credentials being compromised.[7]

In order to use OAuth 2.0, the installed application must either have access to the system browser, or it must have a browser embedded as a web view. The OAuth 2.0 flow for installed applications is as follows:

1. Your application redirects a browser to a Google URL. The URL query parameters indicate the type of Google API access that the application requires.

2. As in other scenarios, Google handles user authentication and consent, and the result of the sequence is an authorization code. The authorization code is returned in the title bar of the browser or as a query string parameter, depending on the parameters your application sends in the request.

3. Your application exchanges the authorization code for an access token and a refresh token. During this exchange, the application presents its client ID and client secret that is obtained from the Developers Console.

4. Your application uses the access token to make calls to a Google API and stores the refresh token for future use.[2]

As we can see from OAuth web flow (figure 0.3), authorization process goes in two steps, I need to somehow do it in a way that is convenient for a user. I decided to start this process by pressing a button, request token will be formed automatically and sent to Google. Second task is to figure out how to get authorization code to our application in order to exchange it for an access token later. One way is to make user to enter authorization code to the application (that will be displayed in browser window after user provide his/her credentials and agrees that our plug-in will use profile information), second way is to get this code as part of the query string that is sent to localhost.[2]

Parameters that are interesting for us to form request token:

redirect_uri Determines where the response is sent. The value of this parameter must exactly match one of the values that appear in the Credentials page in the Google Developers Console (including the http or https scheme, case, and trailing slash). You may choose between urn:ietf:wg:oauth:2.0:oob, urn:ietf:wg:oauth:2.0:oob:auto, or an http://localhost port. Value of this parameter will determine the way of getting authorization code that we will exchange for an access token.

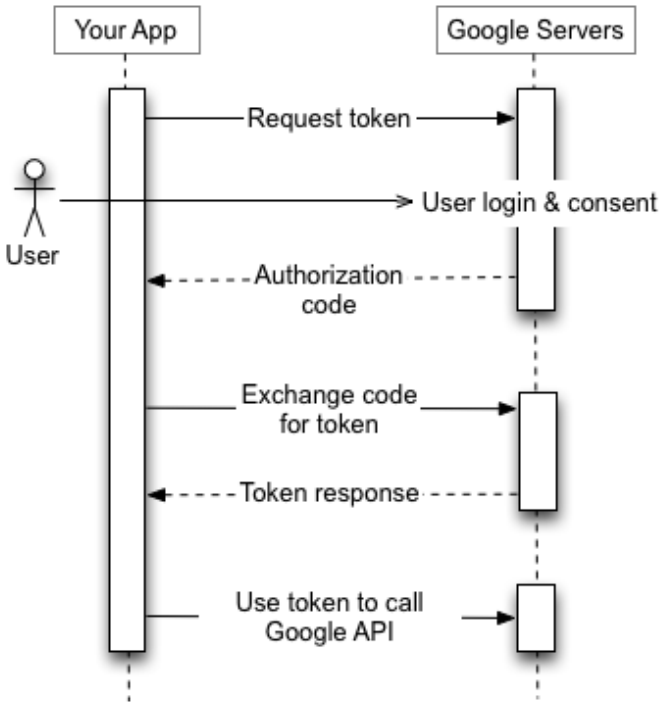

Figure 0.3: OAuth web flow

client_id Identifies the client that is making the request. The value passed in this parameter must exactly match the value shown in the Google Developers Console.

Paradigm shift

At first glance we thought that Creative Suite version of Adobe Premiere Pro with provided SDK will be enough to implement desired functionality of the plug-in, but we encountered a problem with operating multiple files in project bin. There is no possibility to realize correct importing of project files into Adobe Premiere Pro CS5.5, so we decided to completely shift our paradigm and choose different approach to problem solving. We understand that we lose the flexibility of C language and possibility of usage many different frameworks and libraries, but in order to realize our needs it is necessary.

Adobe Premiere Pro CC

Creative Cloud (CC) version of Adobe products is the successor to Creative Suite that we chose as our working environment before encountering a problem. Adobe Premiere Pro CC allows us to use Panel SDK that provides much more functionality in comparison with SDK for Creative Suite version of video editing tool.

Adobe announced that it will no longer release any products of CS version[1], so choosing CC we also looking to the future.

Adobe Premiere Pro CC 2014 uses built-in web browser for executing plug-ins, so that means that we can use standard tools for web development to implement our plug-in.

ExtendScript

ExtendScript is a programming language, that is used for scripting Adobe products. We will use it as a layer between our plug-in and Adobe Premiere Pro since it can operate functions of this software directly. It has very similar structure and syntax with JavaScript because both of these programming languages have common ancestor - ECMAScript.

ECMAScript

This Ecma Standard is based on several originating technologies, the most well known being JavaScript (Netscape) and JScript (Microsoft). The language was invented by Brendan Eich at Netscape and first appeared in that companys Navigator 2.0 browser. It has appeared in all subsequent browsers from Netscape and in all browsers from Microsoft starting with Internet Explorer 3.0.

ECMAScript is an object-oriented programming language for performing computations and manipulating computational objects within a host environment. ECMAScript is not intended to be computationally self-sufficient; indeed, there are no provisions in this specification for input of external data or output of computed results. Instead, it is expected that the computational environment of an ECMAScript program will provide not only the objects and other facilities described in this specification but also certain environment-specific host objects, whose description and behaviour are beyond the scope of this specification except to indicate that they may provide certain properties that can be accessed and certain functions that can be called from an ECMAScript program.

A scripting language is a programming language that is used to manipulate, customise, and automate the facilities of an existing system. In such systems, useful functionality is already available through a user interface, and the scripting language is a mechanism for exposing that functionality to program control. In this way, the existing system is said to provide a host environment of objects and facilities, which completes the capabilities of the scripting language. A scripting language is intended for use by both professional and non-professional programmers.

ECMAScript is object-based: basic language and host facilities are provided by objects, and an ECMAScript program is a cluster of communicating objects. An ECMAScript object is a collection of properties each with zero or more attributes that determine how each property can be used - for example, when the Writable attribute for a property is set to false, any attempt by executed ECMAScript code to change the value of the property fails. Properties are containers that hold other objects, primitive values, or functions. A primitive value is a member of one of the following built-in types: Undefined, Null, Boolean, Number, and String; an object is a member of the remaining built-in type Object; and a function is a callable object. A function that is associated with an object via a property is a method.

ECMAScript defines a collection of built-in objects that round out the definition of ECMAScript entities. These built-in objects include the global object, the Object object, the Function object, the Array object, the String object, the Boolean object, the Number object, the Math object, the Date object, the RegExp object, the JSON object, and the Error objects Error, EvalError, RangeError, ReferenceError, SyntaxError, TypeError and URIError.

ECMAScript also defines a set of built-in operators. ECMAScript operators include various unary operations, multiplicative operators, additive operators, bitwise shift operators, relational operators, equality operators, binary bitwise operators, binary logical operators, assignment operators, and the comma operator.

ECMAScript syntax intentionally resembles Java syntax. ECMAScript syntax is relaxed to enable it to serve as an easy-to-use scripting language. For example, a variable is not required to have its type declared nor are types associated with properties, and defined functions are not required to have their declarations appear textually before calls to them.[4]

NARRA

NARRA API

Design

Adobe Premiere Pro

Overview

Adobe Premiere Pro is a timeline-based video editing software application. It is part of the Adobe Creative Cloud, which includes video editing, graphic design, and web development programs.

Premiere Pro has been used to edit feature films, such as Gone Girl, Captain Abu Raed, and Monsters, and other venues such as Madonna's Confessions Tour.[6]

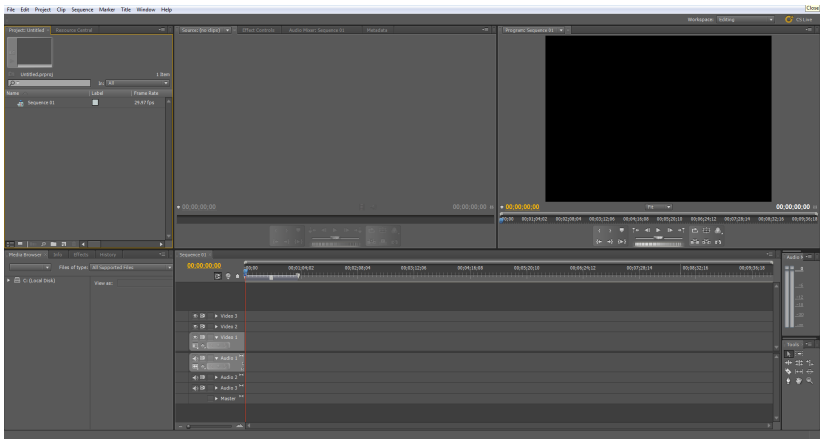

Figure 0.4: Adobe Premiere Pro CS5.5 window

History

Premiere Pro is the redesigned successor to Adobe Premiere, and was launched in 2003. Premiere Pro refers to versions released in 2003 and later, whereas Premiere refers to the earlier releases. Premiere was one of the first computer-based NLEs (non-linear editing system), with its first release on Mac in 1991. Up until version Premiere Pro 2.0 (CS2), the software packaging featured a galloping horse, in a nod to Eadweard Muybridge's work, "Sallie Gardner at a Gallop".[6]

Software Development Kit

The Adobe Premiere Pro plug-in and software development kits (SDK) allow us to build plug-ins to enhance and extend the capabilities of Premiere Pro. I will use it's functionality to implement sequence importer and exporter.

Importer

Importers provide video and/or audio from the media source. This source can be a single file, a set of files, a communication link between another application, etc.

Exporter

Implementation

Conclusion

Bibliography

[1] CNET: Adobe kills Creative Suite, goes subscription-only. 2014, [Online; accessed 30-March-2015]. Available at WWW: <http://www.cnet.com/news/adobe-kills-creative-suite-goes-subscription-only/>

[2] Google: Using OAuth 2.0 for Installed Applications — Google Accounts —- Google Developers. 2014, [Online; accessed 29-March-2015]. Available at WWW: <https://developers.google.com/accounts/docs/OAuth2InstalledApp>

[3] Incorporated, A. S.: Adobe Creative Suite 3 Production Premium Wins in Broadcasting. 2007, [Online; accessed 28-March-2015]. Available at WWW: <https://www.adobe.com/aboutadobe/pressroom/pressreleases/200704/041607BBC.html>

[4] International, E.: ECMAScript Language Specification. 2011, [Online; accessed 30-March-2015]. Available at WWW: <http://www.ecma-international.org/ecma-262/5.1/>

[5] mediasilo: Editing Wars: Adobe Premiere vs Final Cut vs Avid. 2013, [Online; accessed 28-March-2015]. Available at WWW: <http://blog.mediasilo.com/editing-wars>

[6] Wikipedia: Adobe Premiere Pro — Wikipedia, The Free Encyclopedia. 2015, [Online; accessed 26-March-2015]. Available at WWW: <https://en.wikipedia.org/wiki/Adobe_Premiere_Pro>

[7] Wikipedia: OAuth — Wikipedia, The Free Encyclopedia. 2015, [Online; accessed 29-March-2015]. Available at WWW: <https://en.wikipedia.org/wiki/OAuth>

Appendix A

Acronyms

GUI Graphical user interface

XML Extensible markup language

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format