4ME102 Implementing Authentication Service and Access Roles Based Application

|  |
| --- |
| Paulius Davydavičius  School of Computer Science, Physics and Mathematics, Linnaeus University, Sweden  [pd222cz@student.lnu.se](mailto:pd222cz@student.lnu.se) |

**Abstract.** In this report, I describe the modelling diagrams and architectural decisions taken for the 4ME102 assignment no. 2 “Authentication service and access roles”

**Keywords:** Authentication Service, Access Roles, oAuth, openID, API, UML diagrams, modelling software

**INTRODUCTION**

The purpose of this report is to explain the UML diagrams modelled for an authentication service and access roles based application which was developed, explain the architectural decisions and support the choices and assumptions made, present testing scenarios and the limitations. The implementation is accessible at <http://paulius.nl/4ME102A2/>.

**FUNCTIONAL MODEL**

Functional model in UML can be represented with a Use Case diagram (attached as an image file “A2 Use case UML diagram.jpeg”). There is one user type who can acquire three different roles depending on the authentication method chosen.

Below are the descriptions of flows of events for every use case. The use cases are sorted by a position on the homepage of the application.

**Authenticate via Twitter**

Authenticate via Twitter Use Case is initiated when the user clicks on the button Sign In with Twitter. The user is redirected to the Twitter login page. Thereafter the Twitter App Authentication page is shown where s/he can authenticate the website to access personal information from Twitter. If the user is logged on to Twitter on other browser tab or has chosen an option keep him/her logged in to Twitter when s/he was logging in to Twitter previous time, the login page will not be shown.

**Authenticate via Facebook**

Authenticate via Facebook Use Case is initiated when the user clicks on the button Log In (Facebook-look blue button). The user is redirected to the Facebook login page. Thereafter the Facebook App Authentication request page is shown where s/he can authorize the website to access personal information from Facebook. If the user is logged on to Facebook on other browser tab or has chosen an option to keep him/her logged in to Facebook when s/he was logging in to Facebook previous time, the login page will not be shown. The application authentication page is shown only when the user is granting access for the website for the first time. The limitation using Internet Explorer was noticed. After logging out, the user is not automatically redirected to the home page. Therefore an additional options “Log in again” or “Go to the home page” where implemented on the intermediate page.

**Authenticate via Google**

Authenticate via Google Use Case is initiated when the user clicks on the button Sign In With Google. The user is redirected to Google login page. Thereafter the Google App Authentication request page is shown where s/he can authorize the website to access personal information from Google. If the user is logged on to Google on other browser tab or has chosen an option to keep him/her logged in to Google when s/he was logging in to Google previous time, the login page will not be shown. The application authentication page is shown only when the user is granting access for the website for the first time.

**Get YouTube videos based on current location**

Get YouTube videos based on current location Use Case is a role-based use case and is initiated when the user successfully authenticates him/herself via Google. The user is automatically redirected to a page where he is provided with a list of YouTube videos based on his/her current location. As the YouTube API[[1]](#footnote-1) currently does not provide functionality for querying the videos based on location or radius from location (Google Developers Reference Guide), the query is implemented to return video’s by tag. As a value for the tag, the city name of the user location is provided. If the user does not confirm geolocation request, no video’s will be shown.

The use case is functional only within the browsers which support HTML5 geolocation functionality.

**Get Tweets about Växjö**

Get Tweets about Växjö Use Case is a role-based use case and is initiated when the user successfully authenticates him/herself via Twitter. The user is automatically redirected to a page where he is provided with the list of tweets by tag “Växjö”. The Tweets of the last 12 hours will be presented. There is an option to change a keyword for getting the tweets. The keyword can with be hashtag keyword or any keyword contained in a tweet. There is maximum of 200 tweets to be returned. The API returns a maximum of 100 tweets per time. If the keyword is so common that there are more than 200 tweets within the last 12 hours after the last tweet the user will be notified about that. By default or if query text is empty, the tweets about “Växjö” are returned.

**Get Tweets by keyword (no requirement)**

Get Tweets about by keyword Use Case is a role-based use case and is initiated when the user successfully authenticates him/herself via Twitter and enters a keyword for getting the tweets by keyword and clicks the button “Change”. The keyword can with be hashtag based or any keyword contained in a tweet. There is maximum of 200 tweets to be returned. The API returns a maximum of 100 tweets per time. If the keyword is so common that there are more than 200 tweets within the last 12 hours after the last tweet the user will be notified about that. If the query text is empty, the tweets about “Växjö” are returned.

**Like / Share on Facebook Use Case (no requirement, not in the Use-Case diagram)**

The website provides a facebook API built in functionality of liking or sharing about the web-application. This user case is initiated when the user clicks on one of this buttons.

**ANALYSIS**

After analyzing the use cases (which represent the initial requirements), it is clear that the following will be part of the system:

* A boundary object representing the interface between the system main page and the human user (MainPageUI (./index.php))
* A boundary object representing the interface between the system Tweets page (role-based, authenticated via Twitter) and the human user (TweetsPageUI (Twitter/index.php))
* A boundary object representing the interface between the system Flickr images page (role-based, authenticated via Facebook) and the human user (FlickrImagesPageUI (Facebook/index.php))
* A boundary object representing the interface between the system Videos page (role-based, authenticated via Google) and the human user (VideosPageUI (Google/index.php))
* A boundary object representing the interface between the system and Google providing the functionality of authenticating the user and getting geo location related data (Google API)
* A boundary object representing the interface between the system and Facebook providing the functionality of authenticating the user (Facebook API)
* A boundary object representing the interface between the system and Twitter providing the functionality of authenticating the user and querying the tweets (Twitter API: oauth and TwitterAPIExchange)
* A boundary object representing the interface between the system and Flickr providing the functionality of querying the images (Flickr API)
* A boundary object representing the interface between the system and YouTube providing the functionality of querying the videos (YouTube API)
* A session object that keeps user session information (Session)

**OBJECT MODEL**

The application was developed in php and no classes (except for the classes provided by API’s) were used. The class diagram which could represent the development model is not relevant for the implementation. The Sequence and Activity diagrams are described instead.

**Dynamic MODEL**

Dynamic model is represented by 3 sequence and 3 activity diagrams:

* Facebook authentication and Flickr images
  + SD[[2]](#footnote-2) – Facebook authentication.jpg (attached)
  + Activity UML – Facebook.jpg (attached)
* Google authentication and YouTube videos queried by user location (city)
  + SD – Google authentication.jpg (attached)
  + Activity UML – Google.jpg (attached)
* Twitter authentication and tweets by keyword
  + SD – Twitter authentication.jpg (attached)
  + Activity UML – Twitter.jpg (attached)

**TESTING SCENARIO**

The application is a web-based application and due to a large availability of browsers and versions every of which has slightly different or different web semantics interpretation capabilities, the application was thoroughly tested out using

* Different browsers:
  + Google Chrome (Version 38)
  + IE[[3]](#footnote-3) 11
* Different perating Systems
  + Windows 7 Professional
  + Windows 8 Home Edition
  + Android 4.4.2
* Some testing tools
  + IETester

The tests have provided some useful insights on improving the website and making it more user-friendly. I.e. in IE does not redirect the user to hooked URL of the main page when logging off from Facebook, the same page is loaded instead. The additional links to come back to the home page or to login again where added. Some issues when the session (access token) expires have been noticed – when the user tries to authenticate using Twitter, it is being logged off first and the application is being authorized the access only from the second time. As some parts are using HTML5, the most recent browser versions are recommended. For better performance experience Google Chrome is recommended.

There were no tests within iOs with Apple products carried out.

**TECHNOLOGIES**

The following technologies have been used:

* PHP[[4]](#footnote-4)
* Different API’s
* xmlns:fb[[5]](#footnote-5) for using Facebook specific HTML[[6]](#footnote-6) tags
* JS[[7]](#footnote-7)
* JQuery[[8]](#footnote-8)
* JSON[[9]](#footnote-9)
* AJAX[[10]](#footnote-10)
* HTML5

**CONCLUSION**

For the implementation of the system I would use ASP.NET technology. The ASP.NET provides an out-of-the-box functionality of using Model View Controller (ASP.NET MVC 4) architectural design pattern which would be very suitable for the system. The idea of this pattern is layers separation. The system is designed in a way that different controllers take care of manipulating the data (making calls to the models and notifying the views about the changes), views – providing the representation of the data. As the application has a mobile platform support, the MVC pattern allows very easily to implement different views (i.e. mobile view and desktop) for the representation of the same data on different platforms.

If more pages were allowed, Class-responsibility-collaboration cards would have been included in order to represent “the dynamics of object interaction and collaboration” (Wikipedia).

**REFERENCES**

Facebook. *Facebook API*. Available at: <<https://developers.facebook.com/docs/reference/php/facebook-api/>> accessed on: 29 Sept. 2014

Flickr. *The App Garden*. Available at: <<https://www.flickr.com/services/api/flickr.photos.search.html>> accessed on: 4 Oct. 2014

Flickr. *The App Garden*. Available at: <<https://www.flickr.com/services/api/flickr.photos.geo.photosForLocation.html>> accessed on: 4 Oct. 2014

Google. *Developer's Guide: PHP*. Available at: <<https://developers.google.com/youtube/2.0/developers_guide_php>> accessed on: 10 Oct. 2014

Google. *YouTube API v2.0*. Available at: <<https://developers.google.com/youtube/2.0/developers_guide_protocol>> accessed on: 4 Oct. 2014

Linda Lawton. *Google Oauth2 PHP*. Available at: <<http://www.daimto.com/google-oauth2-php/>> accessed on: 3 Oct. 2014

Twitter. *Embedded Tweets.* Available at: < <https://dev.twitter.com/web/embedded-tweets>> accessed on: 8 Oct. 2014

Twitter. *Using OAuth* Available at: <<https://dev.twitter.com/oauth/overview/introduction>> accessed on: 3 Oct. 2014

Twitter community. *Solved Errors Message “Invalid or Expired Token Code”*. Available at: <<https://twittercommunity.com/t/solved-errors-message-invalid-or-expired-token-code-89/10797/9>> accessed on: 8 Oct. 2014

W3. *Main page*. Available at: <<http://www.w3.org/wiki/Main_Page>> accessed on: 10 Oct. 2014

W3schools. *HTML5 Geolocation*. Available at: <<http://www.w3schools.com/html/html5_geolocation.asp>> accessed on: 10 Oct. 2014

WebcodingEasy.com. *Convert Twitter Created\_at time format to ago format*. Available at: <<http://webcodingeasy.com/PHP/Convert-twitter-createdat-time-format-to-ago-format>> accessed on: 8 Oct. 2014

Wikipedia. *XML namespace*. Available at: <<http://en.wikipedia.org/wiki/XML_namespace>> accessed on: 10 Oct. 2014

Wikipedia. *JSON*. Available at: <<http://en.wikipedia.org/wiki/JSON>> accessed on: 10 Oct. 2014

1. Application Programming Interface [↑](#footnote-ref-1)
2. Sequence diagram [↑](#footnote-ref-2)
3. Internet Explorer [↑](#footnote-ref-3)
4. Programming language for developing the web pages [↑](#footnote-ref-4)
5. XML namespace for Facebook [↑](#footnote-ref-5)
6. HTML – hypertext markup language [↑](#footnote-ref-6)
7. Javascript – technology used for developing the web pages to be run on client side [↑](#footnote-ref-7)
8. Javascript based library used in websites development. Runs client side [↑](#footnote-ref-8)
9. “JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.” (Wikipedia) [↑](#footnote-ref-9)
10. Technology, used in web programming, enabling server side calls from client side to be executed asynchronously. [↑](#footnote-ref-10)