Netgen Connect documentation

http://projects.ez.no/ngconnect

https://github.com/netgen/ngconnect

http://www.netgen.hr/eng

Extension version: 1.1

Date: 21.03.2011.

Table of contents

1. Introduction

Netgen Connect is an eZ Publish extension able to provide user sign in by using social network authentication and authorization. (Facebook Connect for example).

At this time, Twitter, Facebook and Tumblr are supported; however, the extension is created in a way that allows additional social networks to be supported just by creating a handler PHP class and adding a reference to it in settings file.

After installing the extension and including its template in your page layout, users are presented with buttons which take them to social network of their choice so they can authenticate themselves and authorize you to access their information. Upon taking an action there (allowing or denying authorization), users are taken back to your eZ Publish installation and automatically logged in to your site in case of successful authorization.

Netgen Connect extension is published under GNU GPLv2 license and as such can be freely used and modified.

2. Installation and configuration

2.1. Installation instructions

Follow the steps outlined below to install the extension.

- Unpack the downloaded package into the *extension* directory of your eZ Publish installation
- Extension requires an additional table to be added to your database. You also need to add the table if upgrading from version 1.0. Use the following command from your eZ Publish root folder, replacing *"user"*, *"password"*, *"host"* and *"database"* with correct values and removing double quotes:

```
mysql -u "user" -p"password" -h"host" "database" <
extension/ngconnect/update/database/mysql/1.1/ngconnect-dbupdate-1.0-to-
1.1.sql
```

- Activate the extension by using the admin interface (Setup -> Extensions) or by appending *ngconnect* to *ActiveExtensions[]* in *settings/override/site.ini.append.php*

```
php bin/php/ezpgenerateautoloads.php --extension
```

  or go to Setup -> Extensions and click the *"Regenerate autoload arrays"* button
- Include *extension/ngconnect/design/standard/templates/ngconnect/ngconnect.tpl* template in place where you want your login buttons to be displayed. It is recommended to include the template somewhere in your page layout to take advantage of using a redirect URL to return the user to the page where he/she clicked the login button
- Clear all caches (from admin 'Setup' tab or from command line)

2.2. Configuration

You can configure this extension by changing the values in *ngconnect.ini*, or, better yet (and recommended way), by overriding *ngconnect.ini* and configure desired settings in your override.

Before using the extension, you need to create Facebook, Twitter or Tumblr apps and have their consumer keys and secrets ready.

You can create the apps on the following links:

Facebook: http://www.facebook.com/developers/

Twitter: https://dev.twitter.com

Tumblr: http://www.tumblr.com/oauth/apps

The following is a list and explanation of all the settings available in ngconnect.ini

- *[ngconnect] LoginMethods* – this array defines all authentication methods available for use. By default, this array contains *facebook*, *twitter* and *tumblr* authentication methods. If you do not want to use some of the methods, comment them out in here. Each of these methods has a *[LoginMethod_**methodname**]* settings group, explained below.
- *[ngconnect] RegularRegistration* (enabled/disabled) - Controls if regular account registration will be enabled or not. If disabled, social account (one that has username, password and email (if not provided by social network) generated randomly) will be created (if needed) and logged in automatically when user signs in with social network, and no choice for regular account creation will be given. If enabled, user is shown a page (*ngconnect/profile*) to enter their regular account details to connect to, or create a new regular account (by setting the username and password himself, and not generate them randomly, thus allowing to login through normal eZ Publish login system). Also, when enabled, user is allowed to skip the page, disable further "nagging" and just create a social account
- *[ngconnect] DuplicateEmailForceRedirect* (enabled/disabled) - This setting controls if regular registration will still be performed when setting above is disabled, unique emails are enabled in the system and user has an email address on social network already present in the system. If enabled, user gets redirected to *ngconnect/profile*. If disabled, a random and invalid email address is generated
- *[ngconnect] LoginWindowType* (page/popup) - Defines how social network login windows will be displayed. If *page* is selected, the login windows will be displayed in the same window/tab that your site is opened in. If *popup* is selected, the login windows will be displayed in browser popup windows
- *[ngconnect] DebugEnabled* (true/false) - Controls if errors in authentication or authorization will be logged to error log
- *[ngconnect] AuthHandlerClasses* – The array that defines authentication handler PHP classes available for each of the supported authentication methods. There's no need to change this setting unless creating a new authentication method
- *[LoginMethod_**methodname**] MethodName* - Name of the authentication method, you can use this to display text on your site or set the alt attribute for login button, for example. This is an universal setting across all *[LoginMethod_*]* settings groups
- *[LoginMethod_**methodname**] DefaultUserPlacement* - Node ID where users using the specific login method will be placed. If not defined, users will be placed to the default user group as specified in *site.ini*. Also universal across all *[LoginMethod_*]* settings groups
- *[LoginMethod_**methodname**] ImageSize* (avatar/original) – Profile image size to fetch from social network. If *avatar* is selected, a 50x50 px profile image will be fetched,

otherwise, the full size image will be fetched. Supported only in *facebook* and *twitter* auth methods

- *[LoginMethod_**methodname**] AppConsumerKey* – Consumer key of your social network app. In *facebook* login method this setting is called *FacebookAppID*
- *[LoginMethod_**methodname**] AppConsumerSecret* – Consumer secret of your social network app. In *facebook* login method this setting is called *FacebookAppSecret*
- *[LoginMethod_facebook] Permissions* – this array defines which permissions will be asked by your app from Facebook, so once user is logged in, you can use those permissions in your app for greater user experience. Specific to *facebook* authentication method

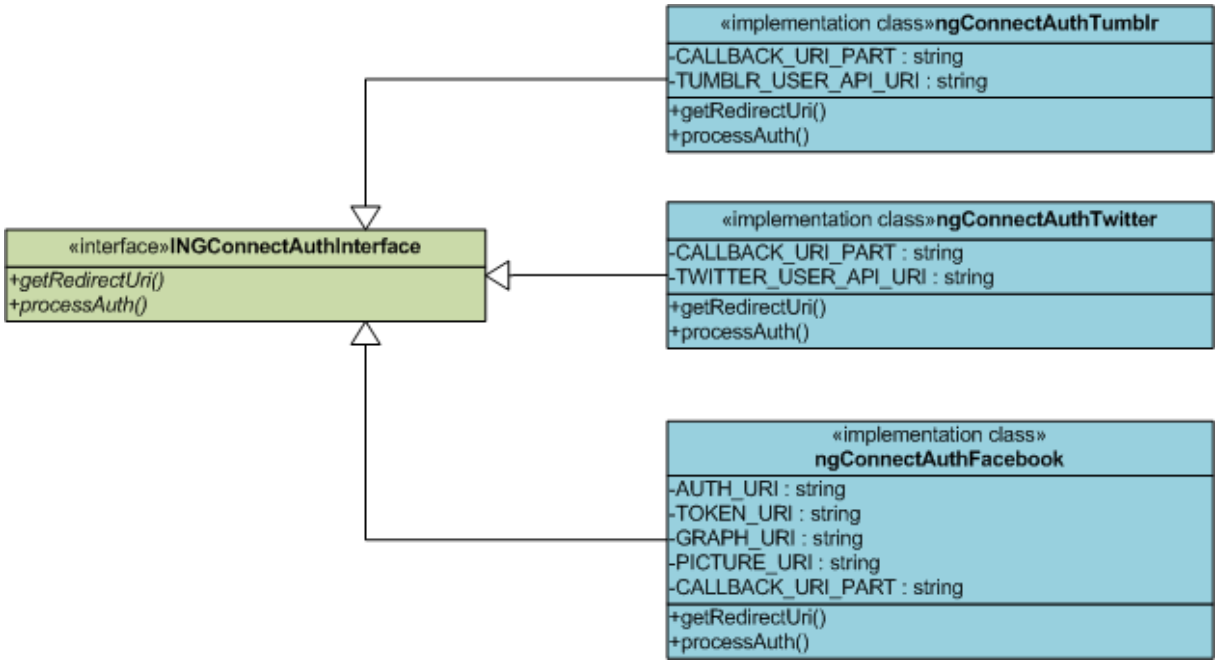
3. Handler development

As already said, Netgen Connect comes with support for three most used social networks, Facebook, Twitter and Tumblr.

If you find yourself in need for supporting additional social networks, you can always create authentication handlers and plug them in with ease.

Developing an authentication handler is a simple process, and consists of creating a PHP class that implements **INGConnectAuthInterface** interface, adding two lines in *ngconnect.ini* (LoginMethods array – defines the name for your social network, and AuthHandlerClasses – defines the name of the PHP class which implements **INGConnectAuthInterface** interface) together with a section in the same ini files with settings specific to your handler ([LoginMethod_*] sections).

The following picture illustrates authentication handlers component of Netgen Connect.



As seen in the picture, **INGConnectAuthInterface** requires that two methods are implemented in your authentication handler class, *getRedirectUri()* and *processAuth()*. Both methods need to be public and static, have zero parameters and need to return an array with the execution result.

getRedirectUri() method is a method that generates a redirection URI that user should be transferred to, to perform authentication and authorization on your social network of choice. Redirection URI needs to be absolute and have a protocol defined, for example "http://www.mynetwork.com/auth?token=12345". The return array needs to look like this:

- In case of successful generation of the redirection URI:

```
return array('status' => 'success', 'redirect_uri' => $myRedirectUri);
```

- In case of unsuccessful generation of the redirection URI:

```
return array('status' => 'error', 'message' => 'Something bad happened!');
```

In this case, 'message' array item is logged in *error.log* if enabled in *ngconnect.ini*.

processAuth() method takes care of processing the result received from social network (validating the response and fetching user info from the social network). Just like *getRedirectUri()* method, this method returns an array which needs to look like this:

- In case of successful validation and fetching of user data:

```
return array('status'           => 'success',
             'login_method'     => 'mynetwork',
             'id'               => $userID,
             'first_name'       => $firstName,
             'last_name'        => $lastName,
             'email'            => $email
             'picture'          => $pictureURI);
```

- 'login_method' item needs to be the same as the name of your auth handler as defined in *[ngconnect] LoginMethods* array in *ngconnect.ini*

- 'id' is required and represents the ID of a user on the social network. It can be any string, but needs to be unique for each user

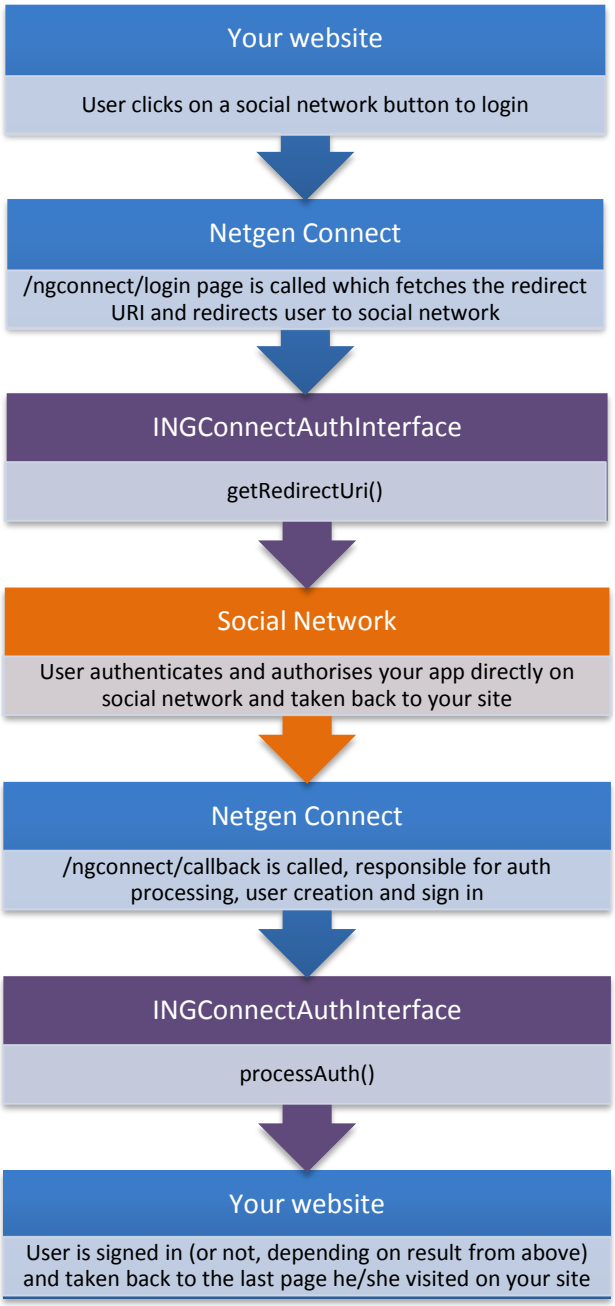
- the rest of the parameters (first name, last name, email, and full URL to user profile picture) are not required and can be empty if no data exists.

- In case of unsuccessful validation and fetching of user data:

```
return array('status' => 'error', 'message' => 'Something bad happened!');
```

In this case, 'message' array item is also logged in *error.log* if enabled in *ngconnect.ini*.

The following is a workflow that shows how exactly Netgen Connect works and as you can see, it's fairly straightforward. In the workflow, blue colors represent your website, along with hardcoded parts of Netgen Connect extension, purple color represents the parts you should be developing when creating a new authentication handler and orange color represents a social network, which you don't have control of.


Your website
User clicks on a social network button to login
Netgen Connect
/ngconnect/login page is called which fetches the redirect URI and redirects user to social network
INGConnectAuthInterface
getRedirectUri()
Social Network
User authenticates and authorises your app directly on social network and taken back to your site
Netgen Connect
/ngconnect/callback is called, responsible for auth processing, user creation and sign in
INGConnectAuthInterface
processAuth()
Your website
User is signed in (or not, depending on result from above) and taken back to the last page he/she visited on your site

4. Important considerations

If you plan to actively use this extension, there are a couple of important considerations that you should be aware of.

We are all well aware of the strengths of eZ Publish content object model. However, in case of this extension, couple of drawbacks come to light. First, extension assumes that the class that is used for user objects creation has an ezuser datatype named "user_account" as per eZ Publish default settings. If not, user will not be able to sign in, nor will User object be created. Also, for now, extension does not support custom login handlers and uses only the standard login handler that comes with eZ Publish.

Since each User object created in the system is a fully fledged user that can sign in, modify their account settings and so on, there's no way to nicely create the users that just want to login to the site without connecting their social network accounts to existing eZ Publish accounts. Because of this, those users are created with a generated username, randomly generated password and an invalid email address (if none is provided by social network). Generated usernames are a combination of string "ngconnect", social network name and user id, i.e. "ngconnect_facebook_12345". Passwords are generated randomly and incompatible password hash type is set as an additional security measure, to prevent guessing of passwords. Email, if not provided by social network, will be stored as invalid and will look like "123456abcdef@localhost.local" where "123456abcdef" is an MD5 hash string.

Considering things said in the previous paragraph and the fact that user, when logged in, can edit their profile, change their passwords and see those generated items, you should make sure that if user has this type of the account, they cannot see and/or modify those aspects of their user profile. For this purpose, a couple of fetch functions are prepared for you to query if current user has an username or email generated by Netgen Connect or if a specific user account (with non generated login details) is connected to a social network by Netgen Connect. Fetch functions are as follows:

- fetch(ngconnect, username_is_generated) - returns true if username is generated by Netgen Connect, false otherwise
- fetch(ngconnect, email_is_generated) - returns true if e-mail is generated by Netgen Connect, false otherwise
- fetch(ngconnect, user_has_connection, hash(user_id, 14, login_method, 'facebook')) - returns true if user with the ID 14 has a connection to social network 'facebook', false otherwise