Pluggable: Facebook

Once upon a time, I had this dream of being able to start up with Facebook-application-development in a way that almost just required "one click", and everything should kickstart. It wasn't that easy of course, but writing on TorneLIB v5 made it possible to at least create a helper API. The helper API is supposed to "kick start" with the most basic things for a Facebook-weblink, so with the help of auth.tornevall.net, that project is going to half-ish-working, to a quite ok API-bridge. As always, it's quite easy to run this one, but since you are actually dependent on a user that approves the usage of Facebook, it's a bit tricky, as you need to redirect urls sometimes.

The API is in a start up phase, so some functions should probably switch to something more stable, as there are stable functions in the SDK. For example reauthorizing an account is possible from the API. We, on the contrary, are using a simple CURL-call for that one, since I never found the function in time.

Initiation

The pluggable facebook-API has its focus on two login-methods: canvas and redirect. The inital version of TornevallWEB had some kind of plan to run in a canvas, so I made two versions - one frameless (redirectLoginHelper) and one for the frames (canvasLoginHelper). For the moment (I haven't checked in several years actually) this part should be auto detecting as soon as you've initializing the plugin. You need this to start:

- The SDK
- Your app id
- Your app secret

The SDK is actually delivered with the API engine, so this is nothing you need to think of, so in short, you only need your credentials. Assuming you have the whole API suite (that makes it possible to run like this), we're running an initialization first:

```
$LIB = new TorneLIB\TorneLIB();
$Facebook = $this->LIB->API->Intialize("facebook", array($FBAppID, $FBAppSecret));
```

So the first thing you have to do after this is to authorize your application, so that you can get a token to run with. For the comfort, you should save this to a database or something, making the token available whenever you need it again.

```
$requestPermissions = array("user_posts","user_managed_groups","user_friends","publish_actions");
$Facebook->Initialize($RedirectURL, $requestPermissions);
$URL = $Facebook->getLoginUrl();

// A lot of actions goes in between here, to mark that the next line is a step after the authorization. This
occurs on a landing page.
if (isset($_REQUEST['code'])) {
    $Facebook->Reauthorize($_REQUEST['code'], $RedirectURL);
}
```

As you can see in the above example, we're adding a few permissions to the authorization request. Those are **appended** to the library defaults **id,name, email**. Following the $URL link should lead you to a Facebook oauth page for approving the premissions. Once done, Facebook will redirect you back to the $RedirectURL. Here's one of the trickies since we need to check if the token must reauthorize or just accepted. For example, if the _GET-parameter **co de** is sent from Facebook, this one is used for an eventual reauthorization of permissions. This is luckily handled by the library. Just keep in mind that the Confirm(), that will be sent back to Facebook, must be identical with the first redirecturl that was used.