Fargo - Development #14147

ajouter un accès oauth2 aux fichiers

29 novembre 2016 18:43 - Frédéric Péters

Statut:	Fermé	**Début:**	29 novembre 2016
Priorité:	Normal	**Echéance:**	04 septembre 2017
Assigné à:	Josué Kouka	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

La spécification faite dans le cadre du projet CUT

branche dev http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2

Demandes liées:

Lié à Fargo - Development #16192: ajouter dépôt de document en oauth2	**Fermé**	**05 mai 2017**

Révisions associées

Révision fe873ff0 - 06 novembre 2017 17:37 - Jean-Baptiste Jaillet

add oauth2 access to get and put a document (#14147)

Historique

#2 - 07 mars 2017 16:03 - Frédéric Péters

- Sujet changé de ajouter un accès oidc aux fichiers à ajouter un accès oauth2 aux fichiers

#3 - 08 mars 2017 09:06 - Mikaël Ates

- Description mis à jour

#4 - 24 mars 2017 09:58 - Jean-Baptiste Jaillet

- Assigné à mis à Jean-Baptiste Jaillet

#5 - 12 avril 2017 16:50 - Jean-Baptiste Jaillet

- Statut changé de Nouveau à Information nécessaire

Alors comme l'a demandé Mick (et je partage l'avis) après plusieurs lectures de doc et spec voici les idées d'implémentation, et les zones de doutes qui persistent.
Pour info,

- la doc indiquée par benj : https://aaronparecki.com/oauth-2-simplified/#other-app-types (que j'ai le plus lue) (1)
- la rfc indiquée par mick : https://tools.ietf.org/html/rfc6749#section-2.3.1
- le cas "pratique" : https://dev.entrouvert.org/projects/grand-lyon-gi/wiki/Sp%C3%A9cification_Interface_porte_document (2)

On part donc sur la partie Web Server Apps de (1).

Ajout d'une app dans fargo (au niveau de fargo premier dossier, c'est bête mais bon autant le placé au bon endroit) : oauth2_api, url en @/oaut2_api/whatever.

La première partie est donc créer ce qui correspond à la première url de (2) /api/get-document/authorize/ avec les paramètres :

- reponse_type=code
- client_id
- redirect_uri
- scope
- state => gardé ça dans un cookie pour vérifié après l'envoi du code si on est bien toujours sur la même requête.

J'avoue que scope je ne vois pas bien son utilité ici, et comment l'utiliser dans fargo. D'après (2) on aurait même besoin uniquement de client_id et redirect_uri.

On arrive donc sur publik à ce moment là pour que l'utilisateur accepte ou pas que l'application est accès à son fichier : basique (allow / cancel), la question c'est est ce que je fais une vue django dans fargo qui redirigera ensuite vers le redirect_uri ? ou une pop up quelque part ? (je suppose

depuis le début que le user est connecté, mais est ce que c'est le cas? est ce que dans le deuxième cas on peut juste faire un @login_required ou c'est plus compliqué ?).

En relisant en fait s'il accepte il choisi un fichier et ensuite on redirige vers le redirect_uri avec le code d'autorisation. En relisant les specs de grand lyon (2) et la doc (1) là j'ai un doute. Il semblerait que d'après (2) on renvoie sur le site du partenaire qui ensuite vient prendre un token puis le fichier. Or, de ce que j'ai compris, dans (1) on récupère ensuite le token puis le fichier et ensuite on finalise. J'ai peut être mal compris un des points.

Bref, quoiqu'il arrive, pour il faut lié le document choisi avec le code de l'utilisateur suite à l'autorisation puis le partenaire demande un token avec le code. Une fois le token donné on le lie aussi au fichier, avec un temps d'éxpiration (je ne sais pas de combien il doit être).
Pour cette partie, il faudrait donc un modèle Oauth2Document:

- une clef étrangère vers un document fargo
- chaine de caractère auth_code
- chaine de caractère token
- champs date token_expiration_date

Enfin le partenaire appelle l'url api/document/get-document avec le token dans le query => on vérifie dans la table du modèle au dessus qu'il y a bien une correspondance avec ce dossier et que le token n'est pas expiré, et on renvoie la document (application/octet-stream avec un header Content-disposition pour le nom du fichier (2)).

Pour le dépôt de document, j'avoue que je ne vois pas le cas : le site partenaire demande un document et l'utilisateur veut en mettre un nouveau mais le choix est fait avant d'être redirigé chez nous ou c'est au moment de la pop-up "accepter / refuser" que l'on propose les deux options?

On doit donc enregistré le document puis renvoyé dans Location l'url d'autorisation d'upload du document où l'utilisateur est redirigé pour accepté (même pop-up). En cas de refus, on récupère le document avec son id et on le supprime (peut être une méthode mieux que de save() puis delete() ?).

En plus de ça, après discussion avec Mick hier, ajouter un modèle d'enregistrement des clients sur l'idée de https://connexion-grandlyon.dev.entrouvert.org/admin/authentic2_idp_oidc/oidcclient/add/.

Donc un modèle Oauth2Client :

- un champs identifiant (chaine de caractère)
- un client_id généré (chaîne de caractère)
- un client_secret généré (chaine de caractère)
- redirect_uri (là je ne sais pas si c'est une liste ou si finalement on n'aura qu'une seule uri de redirection, ce qui me semble être le cas. A confirmer).

Pour redirect_uri comme elle est envoyée dans la première url en paramètre je me demande si elle est nécessaire dans le modèle (A confirmer).
C'est les seuls champs qui me semblent nécessaires, en partant du principe qu'on passe toujours par le même procédé (A confirmer).

Voilà, ça reprends beaucoup de choses déjà écrites, mais ça me permet d'être sûr que je n'ai rien compris de travers.

Comme pour ozwillo, je pensais faire une branche wip/oauth2 dans fargo et mettre les lien de commits ici plutôt qu'une infinité de patch sur le master.

#6 - 18 avril 2017 17:25 - Benjamin Dauvergne

Jean-Baptiste Jaillet a écrit :

> Alors comme l'a demandé Mick (et je partage l'avis) après plusieurs lectures de doc et spec voici les idées d'implémentation, et les zones de doutes qui persistent.
> Pour info,
>
> - la doc indiquée par benj : https://aaronparecki.com/oauth-2-simplified/#other-app-types (que j'ai le plus lue) (1)
> - la rfc indiquée par mick : https://tools.ietf.org/html/rfc6749#section-2.3.1
> - le cas "pratique" : https://dev.entrouvert.org/projects/grand-lyon-gi/wiki/Sp%C3%A9cification_Interface_porte_document (2)
>
> On part donc sur la partie Web Server Apps de (1).
>
> Ajout d'une app dans fargo (au niveau de fargo premier dossier, c'est bête mais bon autant le placé au bon endroit) : oauth2_api, url en @/oaut2_api/whatever.

Il n'y pas d'URL conseillée pour faire des WS, donc gardons celles mises dans la spéc.

> La première partie est donc créer ce qui correspond à la première url de (2) /api/get-document/authorize/ avec les paramètres :
>
> - reponse_type=code
> - client_id
> - redirect_uri
> - scope
> - state => gardé ça dans un cookie pour vérifié après l'envoi du code si on est bien toujours sur la même requête.

Non state on le garde pas dans un cookie vu qu'on l'a dans l'URL, suite à la réponse de l'utilisateur au formulaire qui lui est présenté on renvoie state

comme on l'a reçu.

> J'avoue que scope je ne vois pas bien son utilité ici, et comment l'utiliser dans fargo. D'après (2) on aurait même besoin uniquement de client_id et redirect_uri.

Dans une première implémentation oublions le effectivement.

> On arrive donc sur publik à ce moment là pour que l'utilisateur accepte ou pas que l'application est accès à son fichier : basique (allow / cancel), la question c'est est ce que je fais une vue django dans fargo qui redirigera ensuite vers le redirect_uri ? ou une pop up quelque part ? (je suppose depuis le début que le user est connecté, mais est ce que c'est le cas? est ce que dans le deuxième cas on peut juste faire un @login_required ou c'est plus compliqué ?).

La vue c'est /authorize, c'est elle qui affiche allow/cancel.

> En relisant en fait s'il accepte il choisi un fichier et ensuite on redirige vers le redirect_uri avec le code d'autorisation. En relisant les specs de grand lyon (2) et la doc (1) là j'ai un doute. Il semblerait que d'après (2) on renvoie sur le site du partenaire qui ensuite vient prendre un token puis le fichier. Or, de ce que j'ai compris, dans (1) on récupère ensuite le token puis le fichier et ensuite on finalise. J'ai peut être mal compris un des points.

Moi je dirais qu'on ne lui pose pas la question, on lui affiche directement ses fichiers, il en prend tant mieux c'est "allow", sinon il clique sur le bouton "cancel".

> Bref, quoiqu'il arrive, pour il faut lié le document choisi avec le code de l'utilisateur suite à l'autorisation puis le partenaire demande un token avec le code. Une fois le token donné on le lie aussi au fichier, avec un temps d'expiration (je ne sais pas de combien il doit être).

Non on ne génère un code qu'une fois que le fichier a été choisi sur la page /authorize.

> Pour cette partie, il faudrait donc un modèle Oauth2Document:
>
> - une clef étrangère vers un document fargo
> - chaine de caractère auth_code
> - chaine de caractère token
> - champs date token_expiration_date

Yep, le workflow c'est ça:
1. on redirige l'utilisateur sur /authorize?client_id=etc...
2. on vérifie les paramètres client_id, redirect_url etc.. tout va bien on continue
[si utilisateur pas authentifié -> on l'authentifie, puis on revient en 1. suite au retour de django-mellon]
3. on affiche la liste des documents + un bouton Cancel (on est toujours sur /authorize)

Soit:

4. l'utilisateur choisit un fichier, on retourne sur "redirect_url" avec un paramètre "code" et "state", aussi on stocke un modèle OAuth2Auhotrize qui contient une clé vers le document, une valeur code (celle qu'on a renvoyé) et une valeur access token (celle qu'on renverra) et une date de création
5. l'application appelle le ws /token pour convertir le code en acces token (en s'identifiant avec sont client_id et son client_secret)
6. À l'aide de son access_token l'application s'authentifie + récupère le fichier sur /get-document

4.2 l'utilisateur clique sur Cancel, on renvoie l'erreur OAuth2 authorization_denied (ou un truc du genre)

> Enfin le partenaire appelle l'url api/document/get-document avec le token dans le query => on vérifie dans la table du modèle au dessus qu'il y a bien une correspondance avec ce dossier et que le token n'est pas expiré, et on renvoie la document (application/octet-stream avec un header Content-disposition pour le nom du fichier (2)).

L'access token n'est pas sans la query mais dans un entête Authorization.

> Pour le dépôt de document, j'avoue que je ne vois pas le cas : le site partenaire demande un document et l'utilisateur veut en mettre un nouveau mais le choix est fait avant d'être redirigé chez nous ou c'est au moment de la pop-up "accepter / refuser" que l'on propose les deux options?

C'est décrit à la fin de ma spéc.

#7 - 21 avril 2017 11:52 - Mikaël Ates

Je viens d'ajouter les diagramme de séquence sur
https://dev.entrouvert.org/projects/grand-lyon-gi/wiki/Sp%C3%A9cification_Interface_porte_document.

Pour le dépôt de document, en fin de cinématique, sur quelle URL du Service le porte-document redirige t-il l'utilisateur ?

#8 - 21 avril 2017 13:03 - Benjamin Dauvergne

L'URL qui est passé lors de l'appel à /authorize dans le paramètre redirect_uri.

#9 - 04 mai 2017 02:27 - Jean-Baptiste Jaillet

- Statut changé de Information nécessaire à En cours

http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=3ec505e16eac5b45e5cc3cb62fb2a370bdd080e4

Voilà, finalement j'ai mis les dernière modifications de la recette, et une meilleure implémentation des erreurs.
Il y a juste pour la vue authorize, le paramètre redirect_uri n'est pas obligatoire mais du coup je ne savais pas comment avoir une destination de réponse en son absence.
J'ai testé le HTTP_REFERER mais ça à l'air bancal et dans mes tests l'entête était même absente.
Du coup voilà, je veux bien une piste de ce côté là.

#10 - 04 mai 2017 09:40 - Frédéric Péters

Je ne relis pas tout, je fais juste une méthode, à la Prévert :

```
+    def form_valid(self, form):
+        doc = form.cleaned_data['file_chose']
```

file_chose ce n'est pas terrible du tout comme nom de variable, ça ne veut rien dire.

```
+        authorization = OAuth2Auhotrize.objects.create(user_document=doc)
```

OAuth2Auhotrize, typo dans le nom de la classe.

```
+        getvars = {'code': authorization.code}
+
+        if 'state' in  self.request.GET:
```

Si tu n'arrives pas par toi-même à le voir, passe un pylint sur le fichier, il devrait te pointer qu'il y a deux espaces là.

```
+            getvars['state'] = self.request.GET['state']
+        import pdb; pdb.set_trace()
```

Soupir.

```
+        red_uri = self.request.session['redirect_uri']
```

Personne ne gagne du temps à manipuler des red_uri plutôt que des redirect_uri, nommer correctement.

Bien sûr il faut relire le reste du patch aussi, il doit y avoir mille autres points du même type dedans.

#11 - 04 mai 2017 15:25 - Frédéric Péters

Parce que ça a été repoussé et qu'il y a eu question fantasque sur le sujet.

OAuth2Auhotrize

Auhotrize

Au Ho Tri Ze.

#12 - 04 mai 2017 16:36 - Frédéric Péters

```
+OAUTH2_ERROR = {
+    'unauthorized_client': 'unauthorized_client',
+    'access_denied': 'access_denied',
+    'unsupported_response_type': 'unsupported_response_type',
+    'invalid_request': 'invalid_request',
+}
```

Sert à ?

#13 - 04 mai 2017 16:37 - Frédéric Péters

Si la branche prend en compte les tickets #16118 et #16121, tu peux le noter là et rejeter les tickets ?

#14 - 04 mai 2017 16:47 - Jean-Baptiste Jaillet

Pour le dictionnaire des erreurs en effet, je le retire.

Pour les tickets #16118 et #16121 c'est pris en compte dans la branche. Reste juste la question sur le redirect_uri que j'ai posé lors de mon premier commentaire.

#15 - 04 mai 2017 17:36 - Benjamin Dauvergne

Jean-Baptiste Jaillet a écrit :

> http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=3ec505e16eac5b45e5cc3cb62fb2a370bdd080e4
>
> Voilà, finalement j'ai mis les dernière modifications de la recette, et une meilleure implémentation des erreurs.
> Il y a juste pour la vue authorize, le paramètre redirect_uri n'est pas obligatoire mais du coup je ne savais pas comment avoir une destination de réponse en son absence.

Si c'est obligatoire ici (effectivement la RFC dit que c'est optionnel, pour nous c'est obligatoire).

> J'ai testé le HTTP_REFERER mais ça à l'air bancal et dans mes tests l'entête était même absente.

What ?

> Du coup voilà, je veux bien une piste de ce côté là.

#16 - 04 mai 2017 17:38 - Benjamin Dauvergne

Les tests (je te conseillerai de les écrire en même temps voir avant, ça t'aidera parce que là tu développes en aveugle il me semble).

#17 - 04 mai 2017 17:45 - Jean-Baptiste Jaillet

bon du coup vu avec benj sur jabber, redirect_uri est obligatoire donc c'est ce que j'avais implémenté pour le moment, en cas d'absence de ce paramètre, je renvoie une 400.

#18 - 05 mai 2017 13:05 - Frédéric Péters

Place-t-on la partie "ajout de fichier" dans un autre ticket ?

#19 - 05 mai 2017 14:56 - Jean-Baptiste Jaillet

On peut. J'ai mis les deux sur la branche (dans deux commit séparés, même si historiquement put arrive get).

Ça peut être plus clair en effet.

#20 - 05 mai 2017 15:00 - Jean-Baptiste Jaillet

Pour le dépôt #16192

#21 - 05 mai 2017 15:00 - Jean-Baptiste Jaillet

- Lié à Development #16192: ajouter dépôt de document en oauth2 ajouté

#22 - 08 mai 2017 19:22 - Jean-Baptiste Jaillet

avec les tests. Pour l'instant uniquement déroulement normal
http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=e5cb3a428361f5fd85fb424046149a460a74fa67

#23 - 15 mai 2017 11:30 - Jean-Baptiste Jaillet

Sous le conseil de Fred je déplace le ticket #16250 dans les commentaires ici.

Benj écrivait :

> Ce commentaire concerne le endpoint /get-document/.
>
> On supposera la variable filename contenant le nom du fichier en unicode.

```
> from urllib import quote
> ascii_filename = filename.encode('ascii', 'replace')
> percent_encoded_filename = quote(filename.encode('utf8'), safe='')
> response['Content-Disposition'] = 'attachement; filename="%s"; filename*=UTF-8\'\'%s' % (ascii_filename,
 percent_encoded_filename)
>
```

#24 - 15 mai 2017 11:33 - Jean-Baptiste Jaillet

http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=eba300a0ada715e3dccc8e12a8b3519ddcc96b94

Modifications faites, répercutées sur les tests. Cela ouvre une autre question que je vais rajouter dans le ticket du dépôt (rapport au header Content-disposition).

#25 - 15 mai 2017 11:54 - Frédéric Péters

> "Accepter vous d'ajouter le document"

#26 - 17 mai 2017 11:42 - Frédéric Péters

Plutôt que charger le urls.py général, je préférerais l'include d'un oauth2/urls.py.

#27 - 18 mai 2017 18:41 - Jean-Baptiste Jaillet

Ok prise en compte des remarques.
Pour le commentaires sur 'Accepter vous d'ajouter le document' comme ça concerne la put, je l'ai fait dans l'autre ticket, avec les remarques de benj.

http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=8cd44385cb03a543a10eac9e12e04332a95996e1

#28 - 18 mai 2017 19:28 - Benjamin Dauvergne

Commentaires posés sur le mauvais ticket, je les bouge.

#29 - 19 mai 2017 10:36 - Benjamin Dauvergne

Il faudrait réintégrer les modifications du commit qui suit

- ajout d'un saut de ligne avant OAuth2Client
- ajout d'authenticate_bearer et son utilisation dans get_document
- get_document_token n'utilise pas authenticate_basic (qui en fait pourrait être remplacé par ce code[1] qui couvre les deux cas, authentification HTTP-Basic ou via le POST)

[1] http://git.entrouvert.org/authentic.git/tree/src/authentic2_idp_oidc/views.py#n272

#30 - 21 mai 2017 18:41 - Jean-Baptiste Jaillet

Pris en compte et poussé.
http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=9984d9a532146a70ff546365374d082c108e60ed

#31 - 21 mai 2017 23:48 - Thomas Noël

Relecture super rapide:

```
     url(r'^api/', include(router.urls)),
+    url(r'^api', include('fargo.oauth2.urls'))
```

ça pose pas de soucis, ça ? (peut-être que ça serait mieux de mettre les trucs oauth2 sous une url /oauth2 plutôt que /api ? -- c'est une question et pas une suggestion)

(et sinon, je me demande si on peut mettre un texte du domaine public dans un code GPL ; si c'est possible il faut en signaler la licence particulière dans le COPYING ... ou changer l'exemple en "coincoin", ça sera plus simple)

#32 - 22 mai 2017 10:19 - Jean-Baptiste Jaillet

j'avais justement demander au début parce que dans la spec c'était api/ et il y avait déjà des urls. Benj m'a dit qu'on gardait api, et quand Fred m'a demandé de placer ça dans un include, je suis aller vérifier. Visiblement c'est comme les templates, django regarde toutes les regex pour une url et va chercher dans toutes ces urls celle demandée. S'il n'en trouve pas, caca et par contre s'il en trouve deux pareilles ba caca aussi. Mais dans notre cas il n'y en a pas.

Pour la licence j'ai pas compris : il manque un bout de licence dans le code ou tu veux savoir si certains passages de la licence sont pas bons?

#33 - 22 mai 2017 10:25 - Benjamin Dauvergne

Thomas Noël a écrit :

> (et sinon, je me demande si on peut mettre un texte du domaine public dans un code GPL ; si c'est possible il faut en signaler la licence particulière dans le COPYING ... ou changer l'exemple en "coincoin", ça sera plus simple)

Domaine public = pas de licence = libre de droit, il n'y a rien à dire.

#34 - 22 mai 2017 10:26 - Benjamin Dauvergne

Maintenant on peut quand même citer l'auteur, ça mange pas de pain.

Pas de souci avec l'include, par contre le slash manquant il y a un souci.

#35 - 22 mai 2017 17:52 - Jean-Baptiste Jaillet

hum je viens de vérifier en fait je n'avais pas de licence du tout dans les fichiers (j'étais persuadé de l'avoir fait).
Mais bref, en regardant dans les autres fichiers, j'ai vu qu'on ne mettait pas le texte à chaque fois (pour fargo c'est uniquement dans api_views.py et le .po) du coup je me posais la question : est-ce qu'on doit mettre la licence en haut de chaque fichier ? Juste un fichier de l'app (le views ou modèle) ? Quand Benj parle d'auteur c'est la ligne Entrouvert copyright ou c'est une ligne author: JB Jaillet pour me faire flamber toussa toussa?

En somme, quelles sont nos habitudes ? (j'avais rajouté la licence sur tous presque tous les fichiers pour ozwillo dans hobo par ex)

#36 - 22 mai 2017 18:48 - Jean-Baptiste Jaillet

Ok discuté sur jabber, on parlait du poème pour le test. J'ai rajouté l'auteur et le recueil en haut du fichier.
J'en ai profité pour ajouter la gpl au début des fichiers .py.
Ajout du '/' à la fin du include

http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=4dd3055b4ff994370ce0ade3f5583fd34fa2fc7a

#37 - 24 mai 2017 15:53 - Jean-Baptiste Jaillet

http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=20b12876c1c4ea01b41373808bbf9484d1c5805c

J'ai profité des conseils de #16192 pour les modifications de styles et les changements de domaine dans les tests.

#38 - 26 mai 2017 09:26 - Jean-Baptiste Jaillet

Mise à jour : http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=f7fbf2cca4cff3d7c93cb686f9594db3d7f8b18b

#39 - 26 mai 2017 10:38 - Frédéric Péters

```
+urlpatterns = patterns('',
```

On n'utilise plus patterns, cf #16055.

#40 - 29 mai 2017 11:20 - Jean-Baptiste Jaillet

Pris en compte : http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=6c2b4e06818eebd925a262e8bd185012c92c1b32

#41 - 13 juin 2017 18:05 - Serghei Mihai

Premiere chose qui me saute aux yeux.

Dans fargo/oauth2/views.py

```
            client = OAuth2Client.objects.get(client_id=client_id)
            client.redirect_uri = redirect_uri
            client.save()
```

A chaque appel à /api/get-document/authorize l'url sera stocké dans la base pour le même client oauth2.
Cela peut être une source de conflits quand plusieurs utilisateurs tenteront d'accèder à leurs documents, notamment dans les lignes:

```
        redirect_uri = request.POST.get('redirect_uri', '')
        ....
        if redirect_uri != client.redirect_uri:
            raise OAuth2Exception('invalid_request')
```

Par convention, les urls doivent toutes finir par un /.

#42 - 13 juin 2017 18:15 - Serghei Mihai

Etrange façon de lever les exceptions:

```
        except KeyError:
            uri = self.error_redirect(redirect_uri, 'invalid_request')
            return HttpResponseRedirect(uri)
        except OAuth2Client.DoesNotExist:
```

```
            uri = self.error_redirect(redirect_uri, 'unauthorized_client')
            return HttpResponseRedirect(uri)
        except OAuth2Exception:
            uri = self.error_redirect(redirect_uri, 'unsupported_response_type')
            return HttpResponseRedirect(uri)
```

Pourquoi ne pas vérifier au plus tôt que toutes les clés nécessaires de request.GET soient présentes?
Au lieu de lever l'exception OAuth2Exception pour l'intercepter plus bas, rediriger directement l'usager avec le message d'erreur. Cela laisse nous laisse avec un try... except proche de la récuperation de l'objet OAuth2Client.

#43 - 28 juin 2017 18:55 - Jean-Baptiste Jaillet

remarques mises à jour : http://repos.entrouvert.org/fargo.git/commit/?h=wip/oauth2&id=12e283019887b0649cb5739ba5d192a528d59d0c

#44 - 13 juillet 2017 18:26 - Mikaël Ates

Il manque la déclaration dans l'admin des urls de callback qui peuvent être passées en redirect_uri. Ces Urls peuvent être différentes pour le dépôt et la récupération de document. Donc il manque aussi la vérification de ces URLs dans le traitement des requêtes.

#45 - 21 juillet 2017 18:08 - Josué Kouka

- Description mis à jour

#46 - 28 août 2017 20:23 - Frédéric Péters

- Echéance mis à 04 septembre 2017

- Assigné à changé de Jean-Baptiste Jaillet à Josué Kouka

#47 - 07 septembre 2017 15:43 - Josué Kouka

- Fichier 0002-add-redirects-uris-field-checking-14147.patch ajouté

- Fichier 0001-fix-pep8.patch ajouté

- Patch proposed changé de Non à Oui

Mikaël Ates a écrit :

> Il manque la déclaration dans l'admin des urls de callback qui peuvent être passées en redirect_uri. Ces Urls peuvent être différentes pour le dépôt et la récupération de document. Donc il manque aussi la vérification de ces URLs dans le traitement des requêtes.

Corrigé avec ce patch

#48 - 07 septembre 2017 16:03 - Frédéric Péters

```
fargo/oauth2/migrations/0001_initial.py        | 10 ++++++-
fargo/oauth2/migrations/0002_oauth2tempfile.py | 23 ---------------
```

Tu peux regrouper le tout dans la branche wip/oauth2, avec un seul commit ?

#49 - 07 septembre 2017 16:23 - Josué Kouka

Frédéric Péters a écrit :

> [...]
>
> Tu peux regrouper le tout dans la branche wip/oauth2, avec un seul commit ?

Yep done.

#50 - 21 septembre 2017 15:10 - Josué Kouka

Up !!

#51 - 30 octobre 2017 14:57 - Josué Kouka

- Fichier 0001-add-oauth2-access-to-get-and-put-a-document-14147.patch ajouté

Un patch avec revue pep8, correction de bug sur le controle des urls de redirections

#52 - 06 novembre 2017 12:03 - Benjamin Dauvergne

La demande de mike étant traitée pour les redirect_uri c'est ok pour moi.

#53 - 08 novembre 2017 10:23 - Josué Kouka

- Statut changé de En cours à Résolu (à déployer)

#54 - 06 mars 2018 12:13 - Benjamin Dauvergne

- Statut changé de Résolu (à déployer) à Fermé

Fichiers

0001-fix-pep8.patch	1,21 ko	07 septembre 2017	Josué Kouka
0002-add-redirects-uris-field-checking-14147.patch	8,91 ko	07 septembre 2017	Josué Kouka
0001-add-oauth2-access-to-get-and-put-a-document-14147.patch	32,1 ko	30 octobre 2017	Josué Kouka