Document généralement obsolète.

Familles d'API Passerelle

datasources

Interro d'une sources de données

Modèle de base BaseDataSource : title, slug, description. Renvoie juste une exception NonImplementedError...

api

Requête :

```
GET /data/<slug>/json[?q=...] HTTP/1.0
```

Retour :

```
{
    data: [ { id: "14114", text: "BESSIN" }, ... ],
    err: 0
}
```

Même interrogration possible avec /jsonp à la place de /json dans la requête, le retour est alors en jsonp.

modèle dérivés

- CsvDataSource(BaseDataSource) : + csv_file, key_column (id), value_columns (text)
- DistantCsvDataSource(BaseDataSource) : + url, key_column (id), value_columns (text)

Peut-être un modèle à ajouter : CacheDataSource avec un timeout en secondes, qui servirait de base à DistantCsvDataSource et autres.

repost

Techniques génériques pour effectuer un POST vers une URL distante

Modèle de base BasePost : title, slug, description, url, timeout. Il reposte directement les données reçues dans l'URL indiquée.

api

```
POST /post/<slug>/json[?...]

<objet JSON> ou null
```

Retour :

```
{
    data: ...,
    err: 0
}
```

modèles dérivés

- MakoPost : + mako_template_input, mako_template_output, mako_template_url -- transforme les données reçues (get et post) via un template mako avant de faire le POST (puis template sur le retour, mais s'il n'est pas en json... ?)

register

Abonnement à une ressource.

api

Demande la liste des ressources auxquelles on peut s'abonner, et via quel transport (mail, sms, rss, ...).

Si un utilisateur (user) est précisé dans l'URL, l'api indique pour chaque ressource par quels transports il y est abonné.

Note : le "user" indiqué permet à Passerelle de renvoyer l'info nécessaire au «vrai» système d'abonnement, on peut par exemple imaginer user=benji@trouvert.org;0612345678 pour un système qui propose mail et sms.

Requête :

```
GET /register/<slug>/[?user=...] HTTP/1.0
```

Retour :

```
{
    data: [  # liste de ressources
        {
            name: '...', # nom de la ressource, si possible de type "id" utilisable dans une URL R
EST
            description: '...', # optionnel, description de la ressource
            url: '...', # optionnel, page "plus d'information"
            rss: '...', # optionnel
            transports: {
                available: ['mail', 'rss', 'sms', ...],
                defined: ['mail', 'rss'],  # abonnement du user, si spécifié
            }
        }, ...
    ],
    err: 0
}
```

Mise à jour des abonnements de user à une ou plusieurs ressources :

Requête :

```
POST /register/<slug>/?user=... HTTP/1.0

[ # liste de ressources
    {
        name: '...',
        transports: ['mail', 'rss']  # mode d'abonnements choisis
    }, ...
]
```

Retour :

```
{
    data: {
        message: 'Vous êtes bien abonné à ...',
    },
    err: 0
}
```

message

Permet l'envoi d'un message à une cible (SMS, mail, ...)

api

Envoi :

```
POST /message/<slug>/ HTTP/1.0
```

```
{
   message: '...',
   from: '...',
   to: ['...', '...', ...]
}
```

Retour :

```
{
    data: {
       message: '4 message envoyés',
    },
    err: 0
}
```

Contrôle d'accès

- soit par apikey (fournie dans le GET ou en Header)
- soit par signature compatible portail-citoyen (hmac)

Pour chaque point d'accès :

- many-to-many avec des apiuser
- modèle apiuser :
    - origine / nom
    - clé
    - type de clé : apikey ou clé de signature portail-citoyen (hmac)