Passerelle - Development #25590

Permettre à un @endpoint de préciser ce qu'il attend en POST (json)

07 août 2018 10:09 - Frédéric Péters

Statut:	Fermé	**Début:**	07 août 2018
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Dans contrib/iws/ Emmanuel a introduit l'utilisation de jsonschema, ça devrait être intégré de manière native à @endpoint.

```
@endpoint(
    methods=['post'],
    accept_schema={"$schema": "...", "title": ..., "properties": ...},
)
```

et ça ferait la validation et appellerait le endpoint avec le contenu parsé dans un paramètre nommé "data".

Alternativement, parce que beaucoup d'appels vont être simples (pour pouvoir être faits depuis w.c.s.), il y aurait aussi moyen de partir de ce qui existe déjà pour les GET,

```
@endpoint(
    methods=['post'],
    accept_schema={  # à la place de "parameters"
        "firstname": {
             "description": "First Name",
        },
    }):
```

De ça on créerait la structure json_schema pour la validation. ($schema, $id sur la forme https://passerelle.entrouvert.org/<connecteur>/<endpoint>, title avec le verbose_name du connecteur, type=object, properties=accept_schema).

Demandes liées:

Lié à Passerelle - Development #25432: Étendre les possibilités de documentat...	**Nouveau**	**23 juillet 2018**

Révisions associées

Révision 6c2eeefa - 06 novembre 2018 17:27 - Frédéric Péters

misc: add json schema validation to endpoints (#25590)

Historique

#1 - 07 août 2018 10:09 - Frédéric Péters

- Lié à Development #25432: Étendre les possibilités de documentation des connecteurs ajouté

#2 - 07 août 2018 10:11 - Frédéric Péters

(possibilité aussi de gérer les deux, le schéma intégral dans la situation où il existerait déjà (et on pointerait alors vers celui-ci pour la doc), le schéma généré pour les cas simples et courants)

#3 - 07 août 2018 11:02 - Benjamin Dauvergne

On devrait s'inspirer d'OpenAPI, normalement ils séparent les chemins des actions (GET, POST, PUT, etc..) et pour chaque action on éventuellement un request_body (décrit selon son type mime, souvent application/json) et des paramètres (pouvant venir du chemin si templaté, d'un header, de la query string ou d'un cookie), ça me semble coller.

https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.1.md#operationObject

À noter les paramètres peuvent être décrit au niveau du chemin ou au niveau de l'action (sous-niveau), on pourrait reprendre tel quel le schéma pour l'objet "Path" d'OpenAPI (on aura pas trop de mal à générer du swagger/openapi complet à terme.).

Bien sûr on ne casserai pas l'API endpoint actuelle, juste on y ajouterai un paramètre "openapi", et on y implémenterai que ce qui nous sert en simplifiant, exemple sur DPark:

```
@endpoint(openapi={
    'path': '/address-eligibility/',
    'get': {'description': _('Verify eligibility of an address')},
    'parameters': [
        {
          'name': 'address_sticode',
          'in': 'query',
          'description': _('Address STI code'),
          'required': True,
          'schema': {'type': 'string'},
        },
       {
          'name': 'address_zipcode',
          'in': 'query',
          'description': _('Address ZIP code'),
          'required': True,
          'schema': {'type': 'string'},
        },
       {
          'name': 'address_locality',
          'in': 'query',
          'description': _('Address locality'),
          'required': True,
          'schema': {'type': 'string'},
        },
     }})
....
```

On peut générer name et pattern à partir de path et éventuellement d'une regexp filée dans schema, le but c'est d'avoir un chemin de migration vers OpenAPI sans continuer à trop inventer autour.

Autre exemple pour un POST toujours du DPark:

```
@endpoint(openapi={
    'path': '/register/',
    'post': {
       'description': _('Register a subscription application'),
       'request_body': {
           'schema': {
                'application/json': REGISTER_SCHEMA <-- ici on pourrait simplifier et mettre directement le sc
héma dans request_body ou mettre un clé schema directement dans le dico de "post"
           }
       }
    }
```

En plus de documenter on pourrait aussi directement implémenter la validation des requêtes avec les différents schémas.

#4 - 07 août 2018 11:13 - Frédéric Péters

On a des paramètres à @endpoint qui seraient aujourd'hui incompatibles ? Mon idée derrière étant que si on part dans cette direction à un moment on voudra @endpoint(path='...', 'post': ...) (dégager le openapi), et que si c'est jouable dès maintenant d'avoir ainsi les paramètres à la racine, on devrait directement commencer ainsi.

Aussi, j'aimerais voir dans quelle mesure ça impose des trucs qu'on va trouver redondant à mentionner, genre :

```
          'in': 'query',
          'required': True,
          'schema': {'type': 'string'},
```

Aujourd'hui on crée ce qu'on peut par introspection (genre la liste des paramètres, s'ils sont obligatoire ou pas), et je trouve que c'est quelque chose d'utile à conserver.

#5 - 07 août 2018 11:40 - Benjamin Dauvergne

Frédéric Péters a écrit :

> On a des paramètres à @endpoint qui seraient aujourd'hui incompatibles ? Mon idée derrière étant que si on part dans cette direction à un moment on voudra @endpoint(path='...', 'post': ...) (dégager le openapi), et que si c'est jouable dès maintenant d'avoir ainsi les paramètres à la racine, on devrait directement commencer ainsi.
>
> Aussi, j'aimerais voir dans quelle mesure ça impose des trucs qu'on va trouver redondant à mentionner, genre :

[...]

On pourrait facilement imposer des valeurs par défaut non prévue par OpenAPI et en sortie avoir ce qu'il faut, typiquement pour les paramètre dire que c'est par défaut dans la query et de type 'string', si on veut plus on l'écrit explicitement.

> Aujourd'hui on crée ce qu'on peut par introspection (genre la liste des paramètres, s'ils sont obligatoire ou pas), et je trouve que c'est quelque chose d'utile à conserver.

On peut effectivement mélanger les deux de l'explicite et de l'implicite (je prend un autre exemple) et garder la forme actuelle:

```
@endpoint(name='payment-info', perm='can_access', pattern=r'^(?P<nameid>\w+)/$',
def payment_info(self, request, nameid, filenumber=None):
```

- De name et pattern on génère path='/payment-info/{nameid}/' (ça impose un peu de discipline sur pattern mais pas trop)
- De nameid et du pattern on génère que nameid est un paramètre de path, de type string par défaut et requis
- De filenumber=None on génère que filenumber est un paramètre de query de type string
- Plutôt que du OpenAPI pure sous forme de dico on pourrait se permettre un truc comme cela:

```
@endpoint(...
     parameter__filenumber__description=_('File number'),
     parameter__filenumber__schema__pattern='^[0-9]+$')
...
```

Ici le fait d'avoir schema directement sous la définition d'un paramètre n'est pas OpenAPI mais derrière on replacera la chose au bon endroit (en disant qu'implicitement tout paramètre accepte un schéma JSON)

Pour le body d'un POST en jons on pourrait faire un truc comme:

```
@endpoint(....
     json__field1__description=_(...),
     json__field1__required=_(...),
)
```

L'idéal c'est d'accepter du pure OpenAPI et des raccourcis quand ça nous arrange, je suis plus embêter pour les méthodes, mais si on s'impose que des paramètre doivent toujours être valides en POST et GET pour un endpoint qui acceptent les deux ça devrait pas trop nous gêner (request_body/json__ impose la génération d'une opération POST, parameters est toujours au niveau du PATH, si pas de methods on génère toujours du GET et rien d'autre).

#6 - 06 novembre 2018 17:28 - Frédéric Péters

- Fichier 0001-misc-add-json-schema-validation-to-endpoints-25590.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Pour commencer, éviter la duplication dans les connecteurs de validation de schéma, quelque chose comme ça.

#7 - 07 novembre 2018 20:57 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

Ack.

#8 - 09 novembre 2018 18:49 - Emmanuel Cazenave

Dans #27653, rebasé ma branche sur celle-ci et adapté mes endpoints, ça fait le taf.

#9 - 12 novembre 2018 16:50 - Frédéric Péters

- Statut changé de Solution validée à Résolu (à déployer)

```
commit 6c2eeefaaf73bfc133269e5963a110658425f2fa
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Tue Nov 6 17:25:11 2018 +0100

    misc: add json schema validation to endpoints (#25590)
```

#10 - 03 décembre 2018 15:14 - Benjamin Dauvergne

- Statut changé de Résolu (à déployer) à Fermé

Fichiers

0001-misc-add-json-schema-validation-to-endpoints-25590.patch	4,56 ko	06 novembre 2018	Frédéric Péters