Développement d'une cellule alimentée par JSON

Il est possible d'ajouter de nouveaux types de cellules pour afficher des données disponibles au format JSON, ces cellules apparaitront ensuite dans la liste des cellules disponibles, sous un sous-titre "Extra".

Définition de la cellule

Dans les *settings* (soit le settings.py du projet, soit le settings.json du domaine), la définition se fait dans une clé JSON_CELL_TYPES.

Par exemple, pour la création d'une cellule affichant la qualité de l'air, tirant ses données du site strasmap.eu, la cellule serait annoncée de la sorte :

```
"JSON_CELL_TYPES": {
  "info-atmo": {
    "name": "Info Atmo",
    "url": "http://strasmap.eu/remote.amf.json/Atmo.status"
  }
}
```

Rendu de la cellule

La suite du travail consiste à fournir un gabarit de rendu pour ces données, ceux-ci se placent dans le répertoire templates/combo/json/ du site (e.g. /var/lib/combo/tenants/www.example.net/templates/combo/json/), avec comme nom l'identifiant de la cellule + ".html", ex: info-atmo.html.

Il s'agit d'un gabarit Django, ils sont documentés en détails à cette adresse : https://docs.djangoproject.com/fr/1.11/ref/templates/

Les données qui auront été téléchargées sont mises à disposition dans la variable json; ainsi, avec le fichier téléchargé ayant cette structure :

```
{
   "cdate" : "mer. 06 déc. 2017",
   "calltime" : 10000,
   "datatime" : "mer. 06 déc. 2017 à 09:49",
   "pdate" : "jeu. 07 déc. 2017",
   "ctext" : "
Pour ce mercredi, le maintien d'un temps calme accompagné de brumes matinales favorisera une nouve
lle augmentation des niveaux de particules.
             La qualité de l'air devrait être bonne à moyenne en plaine avec des indices 4 à 5 su
r 10 en particules alors qu'elle restera très bonne sur les reliefs
             avec des indices 2 en ozone.",
   "ptext" : "
Jeudi, sous un temps sec et lumineux, les émissions de polluants seront bien dispersées par le ven
t. En plaine, les particules devraient se contenir
             à des indices 4 sur 10 maximum et sur les reliefs l'ozone dominera à des indices 2-3
. C'est donc une bonne voire très bonne qualité de l'air qui
             sera attendue sur le territoire alsacien.",
   "s" : [
      {"lc" : "bonne", "cp" : "#9AEB00", "cc" : "#9AEB00", "sp" : "status_4", "lp" : "bonne", "sc"
 : "status_4", "id" : 67043},
      {"lc" : "bonne", "cp" : "#1AD100", "cc" : "#1AD100", "sp" : "status_3",   "lp" : "bonne", "
sc" : "status_3", "id" : 67049},
      […]
```

La cellule doit trouver la ligne correspondant au code INSEE de la commune pour en afficher la qualité de l'air puis afficher

l'information textuelle générale.

```
{% for city in json.s %}
  {% if city.id == 67482 %}<h2>Qualité de l'air : {{city.lc}}</h2>{% endif %}
{% endfor %}
<p>{{json.ctext}}</p>
<p><small>{{json.datatime}}</small></p>
```

Paramétrage de la cellule

Plutôt qu'avoir le code INSEE en dur dans le gabarit, il est possible de rendre la cellule paramétrable en faisant évoluer sa déclaration vers la suivante :

```
"JSON_CELL_TYPES": {
  "info-atmo": {
    "name": "Info Atmo",
    "url": "http://strasmap.eu/remote.amf.json/Atmo.status",
    "form": [
      {
        "varname": "commune",
        "type": "string",
        "label": "Commune"
      }
    ]
  }
}
```

Les paramètres sont accessibles dans la variable parameters, le gabarit devient donc :

```
{% for city in json.s %}
  {% ifequal parameters.commune city.id|stringformat:"s" %}<h2>Qualité de l'air : {{city.lc}}</h2>
{% endif %}
{% endfor %}
<p>{{json.ctext}}</p>
<p><small>{{json.datatime}}</small></p>
```

Dans cet exemple précis, le code INSEE de la commune se trouve sous forme d'entier et non de chaine de caractères, la condition utilise donc {% ifequal %} et city.id est converti en chaine par le filtre stringformat:"s".

Récupération de données supplémentaires

La suite, c'est se dire qu'il est plus facile d'écrire Oberschaeffolsheim que 67350 et permettre le paramétrage de la cellule via un nom de commune plutôt que le code INSEE. Pour réaliser ça la cellule va faire appel à une deuxième source de données, qui contient la correspondance entre les codes et les noms.

La déclaration de la cellule ajoute donc cette seconde URL :

```
"JSON_CELL_TYPES": {
  "info-atmo": {
    "name": "Info Atmo",
    "url": "http://strasmap.eu/remote.amf.json/Atmo.status",
    "additional-data": [
      {"key": "communes",
       "url": "http://strasmap.eu/remote.amf.json/Atmo.geometry"
      },
    ],
    "form": [
```

```
      {
        "varname": "commune",
        "type": "string",
        "label": "Commune"
      }
    ]
  }
}
```

Le contenu qui en sera récupéré sera mis à disposition dans la variable communes (spécifiées dans l'attribut key). Il s'agit donc de boucler pour trouver le code de la commune selon son nom puis de reprendre comme précédemment sur la recherche de la qualité associée au code :

```
{% for commune in communes.s %}
  {% if commune.ln == parameters.commune %}
    {% for q in json.s %}
      {% ifequal commune.id q.id|stringformat:"s" %}
        <h2>Qualité de l'air : {{q.lc}}</h2>
      {% endifequal %}
    {% endfor %}
  {% endif %}
{% endfor %}
<p>
{{json.ctext}}
</p>
<p><small>{{json.datatime}}</small></p>
```

Gestion du cache

Par défaut les données téléchargées sont conservées en cache 60 secondes, on peut augmenter cette durée en accompagnant les URL d'une clé cache_duration :

```
"JSON_CELL_TYPES": {
  "info-atmo": {
    "name": "Info Atmo",
    "url": "http://strasmap.eu/remote.amf.json/Atmo.status",
    "cache_duration": 3600,
    "additional-data": [
      {"key": "communes",
       "url": "http://strasmap.eu/remote.amf.json/Atmo.geometry",
       "cache_duration": 86400
      },
    ],
    "form": [
      {
        "varname": "commune",
        "type": "string",
        "label": "Commune"
      }
    ]
  }
}
```

Utilisation du profil de l'usager

Pour revenir au contenu, plutôt qu'un paramétrage de la commune, on pourrait vouloir la récupérer depuis le profil de l'usager, on passerait alors par un appel supplémentaire, vers l'API du fournisseur d'identités :

```
"JSON_CELL_TYPES": {
  "info-atmo": {
    "name": "Info Atmo",
    "url": "http://strasmap.eu/remote.amf.json/Atmo.status",
    "cache_duration": 3600,
    "additional-data": [
      {"key": "communes",
       "url": "http://strasmap.eu/remote.amf.json/Atmo.geometry",
       "cache_duration": 86400
      },
      {"key": "user",
       "url": "{{ idp_url }}api/users/{{ user_nameid }}/"
      }
    ]
  }
}
```

On voit ici la possibilité d'utiliser des variables dans les URL, {{ idp_url }} reprendra l'adresse du fournisseur d'identités et {{ user_nameid }} contiendra l'identifiant unique de l'usager.

Plutôt que comparer avec le paramètre de la cellule, la comparaison peut donc se faire par rapport au profil de l'usager, pour lequel on suppose qu'il contiendra sa commune dans le champ city :

```
{% for commune in communes.s %}
  {% if commune.ln == user.city %}
    {% for q in json.s %}
      {% ifequal commune.id q.id|stringformat:"s" %}
        <h2>Qualité de l'air : {{q.lc}}</h2>
      {% endifequal %}
    {% endfor %}
  {% endif %}
{% endfor %}
<p>
{{json.ctext}}
</p>
<p><small>{{json.datatime}}</small></p>
```