

Commander l'API de l'Open Data de Nantes avec Processing

Conditions	Travail en groupe Durée : 2h	Moyens	<ul style="list-style-type: none"> • Poste informatique sous Windows • Connexion internet • Processing 1.5 • Notepad++
Prérequis	Etre initié à Processing	Niveau	Classe de Terminale S spécialité ISN
Compétences	<ul style="list-style-type: none"> • Compétences C2 : Concevoir et réaliser une solution informatique en réponse à un problème 		
Eléments du programme	<ul style="list-style-type: none"> • Représentation de l'information <ul style="list-style-type: none"> ○ Formats ○ Structuration et organisation de l'information • Algorithmes <ul style="list-style-type: none"> ○ Algorithmes simples (Rechercher un élément dans un tableau) • Langages et programmation <ul style="list-style-type: none"> ○ Types de données ○ fonctions 		

La ville de Nantes met à disposition un certains nombres de données accessibles directement à l'aide d'une requête transmise sur le serveur de la ville. Toutes les requêtes sont décrites dans la documentation de l'API "Application Programming Interface" du site.

Pour envoyer nos requêtes et récupérer puis traiter les données nous allons utiliser une fois de plus Processing 1.5.

Quelles sont les requêtes possibles sur l'API ? Comment traiter les données transmises par l'API ? Comment visualiser les données ?

Table des matières :

1. Fonctionnement de l'API	2
2. Mise en œuvre de l'API avec Processing.....	2
3. Interprétation des données XML.....	3
4. Mettre en forme les données de l'API	5

1. Fonctionnement de l'API

L'Open Data de Nantes offre deux API pour accéder à ses données. [L'API v1](#) est la première à avoir été mise en place. Elle porte sur les données liées à la mobilité et dont les valeurs sont actualisées en temps réel.

1. Lister les commandes donnant accès aux données en temps réel

- **Commande "getDisponibiliteParkingsPublics"** : Cette commande permet de récupérer les informations concernant la disponibilité des parkings publics de l'agglomération nantaise.
- **Commande "getFluiditeAxesRoutiers"** : Cette commande retourne les indicateurs de fluidité du trafic sur les tronçons routiers.
- **Commande "getInfoTraficTANPrevisionnel"** : Cette commande permet de récupérer l'info trafic prévisionnel des bus et tramway de la SEMITAN.
- **Commande "getInfoTraficTANTempsReel"** : Cette commande permet de récupérer l'info trafic en temps réel des bus et tramway de la SEMITAN
- **Commande "getTempsParcours"** : Indication des temps de parcours en minutes sur chaque itinéraire.

2. Indiquer le format d'une requête sur l'API

<http://data.nantes.fr/api/<cmd>/<version>/<key>>

2. Mise en œuvre de l'API avec Processing

Pour pouvoir interroger la base de données, il faut créer un compte sur le site de l'Open Data de Nantes. Puis obtenir une clé d'application en remplissant un petit formulaire (voir séquence n°1).

Avec votre clé d'application créée, nous allons écrire un petit programme avec Processing **1.5**. Ce programme va venir se connecter au service pour l'interroger à l'aide d'une adresse internet (ou **URL**) constituée de plusieurs éléments dont notre clé et la commande de la requête. Il faut aussi indiquer le numéro de version de l'API utilisée.

La requête s'effectue en une seule instruction avec Processing : **loadStrings**. Cette dernière nous renvoie un tableau de chaîne de caractères (String), chaque entrée de ce tableau correspondant à une ligne de la réponse donnée par l'API de la ville de Nantes. Cette réponse nous est fournie au format XML et nous allons dans un premier temps simplement l'imprimer sur la console de Processing.

```
String datanantesCle = "309NRRB5VI21JW6";
String datanantesCommande = "getDisponibiliteParkingsPublics";
String datanantesVersion = "1.0";

void setup()
{
  // Adresse à laquelle nous allons accéder aux données
  String url =
  "http://data.nantes.fr/api/"+datanantesCommande+"/"+datanantesVersion+"/"+d
  atanantesCle;

  // Connexion à l'API et chargement des données avec la clé et la commande
  String lines[] = loadStrings(url);
```

```
// Impression du résultat sur la console
for (int i=0;i<lines.length;i++)
    println(lines[i]);
}
}
void draw()
{
    // On ne dessine rien pour l'instant
}
```

3. Tester le programme et reporter le résultat obtenu dans la fenêtre console de Processing :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<opendata>
<request>http://data.nantes.fr/api/getDisponibiliteParkingsPublics/1.0/309N
RRB5VI21JW6</request>
<answer>
<status code="0" message="OK"/>
<data><Groupes_Parking>
    <Groupe_Parking>
        <Grp_identifiant>2</Grp_identifiant>
        <Grp_nom>DECRE-BOUFFAY</Grp_nom>
        <Grp_statut>2</Grp_statut>
        <Grp_pri_aut>0</Grp_pri_aut>
        <Grp_disponible>0</Grp_disponible>
        <Grp_complet>0</Grp_complet>
        <Grp_exploitation>0</Grp_exploitation>
        <Grp_horodatage>04/02/2013 22:44:40</Grp_horodatage>
        <IdObj>284</IdObj>
    </Groupe_Parking>
    .....
    .....
    <Groupe_Parking>
        <Grp_identifiant>34</Grp_identifiant>
        <Grp_nom>GARE SUD 4</Grp_nom>
        <Grp_statut>2</Grp_statut>
        <Grp_pri_aut>0</Grp_pri_aut>
        <Grp_disponible>0</Grp_disponible>
        <Grp_complet>0</Grp_complet>
        <Grp_exploitation>0</Grp_exploitation>
        <Grp_horodatage>04/02/2013 22:44:40</Grp_horodatage>
        <IdObj>3989</IdObj>
    </Groupe_Parking>
</Groupes_Parking></data>
</answer>
</opendata>
```

3. Interprétation des données XML

Plutôt que d'utiliser la fonction `loadStrings`, nous allons utiliser les fonctionnalités de Processing dédiées à l'interprétation du **format XML**. Ainsi seule la fonction d'appel et le format du résultat changent :

```
XMLElement xml = new XMLElement(this, url);
```

4. Modifier le programme précédent et reporter votre résultat :

```
void setup()
{
    // Adresse à laquelle nous allons accéder aux données
    String url =
    "http://data.nantes.fr/api/"+datanantesCommande+"/"+datanantesVersion+"/"+d
    atanantesCle;

    // Connexion à l'API et chargement des données avec la clé et la commande
    XMLElement xml = new XMLElement(this, url);
    // Impression du résultat sur la console
    println(xml);
}
```

Nous constatons la répétition de l'élément *<Groupe_Parking>*, chacun de ces **noeuds** contenant des informations relatives à un parking. Voici un extrait de la réponse :

```
<opendata>
<request>http://data.nantes.fr/api/getDisponibiliteParkingsPublics/1.0/309N
RRB5VI21JW6</request>
<answer>
  <status code="0" message="OK"/>
  <data>
    <Groupes_Parking>
      <Groupe_Parking>
        <Grp_identifiant>2</Grp_identifiant>
        <Grp_nom>DECRE-BOUFFAY</Grp_nom>
        <Grp_statut>2</Grp_statut>
        <Grp_pri_aut>0</Grp_pri_aut>
        <Grp_disponible>0</Grp_disponible>
        <Grp_complet>0</Grp_complet>
        <Grp_exploitation>0</Grp_exploitation>
        <Grp_horodatage>04/02/2013 23:02:40</Grp_horodatage>
        <IdObj>284</IdObj>
      </Groupe_Parking>
    .....
  </data>
</answer>
</opendata>
```

Nous souhaitons récupérer le nombre de parkings dans Nantes, ce nombre étant directement liés au nombre de noeuds *<Groupe_Parking>*. Grâce à Processing qui permet de traiter les données XML, nous allons stocker dans un tableau de XMLElement tous les noeuds *<Groupe_Parking>*. Pour cela, il est nécessaire d'utiliser la fonction *getChildren(path)* et de définir le chemin direct pour accéder aux données de *<Groupe_Parking>*.

5. Compléter le programme suivant :

```
XMLElement[] parkings = xml.getChildren(. . . . .);
println(parkings[1]);
int nombreParking = parkings.length;
println("Nombre de parkings sur Nantes : "+ nombreParking);
```

```
XMLElement[] parkings =
xml.getChildren("answer/data/Groupes_Parking/Groupe_Parking");
println(parkings[1]);
```

```
int nombreParking = parkings.length;
println("Nombre de parkings sur Nantes : "+nombreParking);
```

6. Tester et reporter le résultat :

```
<Groupe_Parking>
  <Grp_identifiant>3</Grp_identifiant>
  <Grp_nom>TOUR DE BRETAGNE</Grp_nom>
  <Grp_statut>5</Grp_statut>
  <Grp_pri_aut>0</Grp_pri_aut>
  <Grp_disponible>268</Grp_disponible>
  <Grp_complet>20</Grp_complet>
  <Grp_exploitation>645</Grp_exploitation>
  <Grp_horodatage>04/02/2013 23:24:40</Grp_horodatage>
  <IdObj>299</IdObj>
</Groupe_Parking>

Nombre de parkings sur Nantes : 23
```

Nous souhaitons récupérer le nom du parking n°1. Pour cela, il vous faut utiliser les fonctions getChild("Grp_nom") et getContent().

7. Modifier votre programme afin d'afficher le nom de parking n°1 (TOUR DE BRETAGNE).

```
XMLElement nomXML = parkings[1].getChild("Grp_nom");
String nom1 = nomXML.getContent();
```

4. Mettre en forme les données de l'API

L'objectif est de mettre en forme les données des parkings en les affichant dans la fenêtre graphique de Processing. Pour cela, vous devez afficher du [texte](#) et gérer les [polices](#) de caractère.

8. Créer un programme affichant toutes les informations du parking n°1 dans la fenêtre de visualisation

Exemple de correction :

```
PFont maFonte;
String datanantesCle = "309NRRB5VI21JW6";
String datanantesCommande = "getDisponibiliteParkingsPublics";
String datanantesVersion = "1.0";
int NombreParkings;
String nom1;

void setup()
{
  // Dimension de la fenêtre
  size(600,600);
  // On charge notre fonte
  maFonte = loadFont("Geneva-15.vlw");
  textFont(maFonte,20);
  // Adresse à laquelle nous allons accéder aux données
  String url = "http://data.nantes.fr/api/"+datanantesCommande+"/"+
datanantesVersion+"/"+datanantesCle;
  // Connexion à l'API et chargement des données avec la clé et la commande
  XMLElement xml = new XMLElement(this, url);
```

```
XMLElement[] parkings = xml.getChildren(
"answer/data/GROUPES_Parking/Groupe_Parking");
NombreParkings = parkings.length;
XMLElement nomXML = parkings[1].getChild("Grp_nom");
nom1 = nomXML.getContent();
}

void draw()
{
    background(255);
    fill(0);
    textFont(maFonte,20);
    text("nombre de parkings sur Nantes :",0,20);
    fill(0,0,255);
    textFont(maFonte,20);
    text(NombreParkings,400,20);
    fill(0);
    text("parking n°1:",0,120,150,25);
    fill(0,0,255);
    text(nom1,150,120,300,25);
}
```