

Ressources naturelles Canada - GéoBase

# Intégration de l'extension CHyF dans la PGF

V.1

Dave Brochu  
10-15-2018

# Table des matières

<b>INTRODUCTION :</b> .....	<b>2</b>
OBJECTIF : .....	2
NOTES IMPORTANTES : .....	2
LIENS UTILES : .....	2
<b>CRÉATION D'UNE EXTENSION :</b> .....	<b>3</b>
CRÉATION DU DOSSIER DE L'EXTENSION .....	3
OBJET GLOBAL POUR LA LIAISON AVEC LA PGF .....	3
CONVERSION DES FICHIERS TYPESCRIPT VERS JAVASCRIPT .....	4
AJOUT D'UNE EXTENSION AU FICHIER HTML DU FGPV-VPGF .....	5
<b>STRUCTURE DES EXTENSIONS</b> .....	<b>6</b>
CLASSE EXTENSION .....	6
CLASSE CHYFEXTENSION .....	7
CLASSE MANAGEEXTENSION .....	7
FLUX DE TRAVAIL .....	7

## Introduction :



### Objectif :

Ce document a pour but de renseigner sur l'ensemble de l'architecture solution pour la réalisation d'extensions. Il relate de la création complète d'une extension jusqu'à son intégration dans l'application de la PGF.

### Notes importantes :

Toutes extensions réalisées pour la PGF doivent être écrites sous Node.js et JavaScript ES6+. Il est fortement recommandé d'utiliser la syntaxe TypeScript qui permet d'améliorer et de sécuriser la production de code JavaScript. Vous devez installer Node.js pour pouvoir utiliser ses fonctionnalités.

**Important :** Il est nécessaire d'obtenir le code source de la PGF. Le dossier de la PGF « **fgpv-vpgf** » et celui des extensions « **extensions** » doivent résider sous un même dossier parent. Vous devez respecter les noms des dossiers.

Name	Date modified	Type	Size
 extensions	2018-10-26 10:11	File folder	
 fgpv-vpgf	2018-10-26 10:25	File folder	

Le code dans ce guide utilise la syntaxe TypeScript qui peut générer des erreurs si vous utilisez JavaScript.

### Liens utiles :

Nom	Répertoires GitHub
PGF	<a href="https://github.com/fgpv-vpgf/fgpv-vpgf">https://github.com/fgpv-vpgf/fgpv-vpgf</a>
Extensions	<a href="https://github.com/davebrochu15/extensions">https://github.com/davebrochu15/extensions</a>

## Création d'une extension :

### Création du dossier de l'extension

Sous une invite de commande :

1. Déplacez-vous vers le nouveau dossier préalablement créé.
2. Entrer la commande pour créer un fichier « **package.json** ». Celui-ci va être généré automatiquement par les valeurs entrées.

```
npm init
```

3. Installer toutes dépendances à l'extension.

```
npm install [-g] [--save-dev] nomExtension
```

4. Si vous utilisez la syntaxe TypeScript, vous devez fournir un fichier de configuration « **tsconfig.json** ». Voici un fichier de configuration de base pour le projet :

```
{
  "compilerOptions": {
    "allowJs": true,
    "baseUrl": "./",
    "target": "ES2016",
    "paths": {
      "api/*": ["../../fgpv-vpgf/api/src/*"],
      "app/*": ["../../fgpv-vpgf/src/app/*"]
    },
    "lib": ["dom", "es6", "es2016", "es2017.object"],
    "noImplicitAny": false,
    "noImplicitThis": true,
    "strictNullChecks": false,
    "allowSyntheticDefaultImports": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
  },
  "exclude": [
    "./node_modules"
  ]
}
```

### Objet global pour la liaison avec la PGF

Vous devez créer un objet global dans la classe de l'extension. Celui-ci a pour but de créer une instance accessible depuis le visualiseur de la PGF. La fonction « init » est une fonction spéciale, appelée par le visualiseur, permettant d'accéder aux cartes de l'api.

```
/**
 * The global extension's name (chyfExtension) used by the RAMP.
 * The init function it called by the RAMP with the maps api
 */

(<any>window).chyfExtension = {
  init: function(api: any) {
    // code
  }
};
```

## Conversion des fichiers TypeScript vers JavaScript

Ajouter les règles de compilation aux configurations de Webpack « **webpack.config.js** ».

1. Ajouter le chemin du fichier.

```
const {PATH} = path.join(SOURCE_PATH, './{dossier});
```

2. Ajouter une entrée sous « entry ».

```
{nomfichier}: path.join({PATH}, './fichier.ts')
```

3. Installer les dépendances des dossiers sous une invite de commande.

```
npm install
```

4. Compiler les fichiers sous une invite de commande.

```
npm run dev
```

5. Vous devriez voir votre fichier « **.js** » sous le dossier « **dist** ».

## Ajout d'une extension au fichier HTML du fgpv-vpgf

Sous « **src/content/samples** », les fichiers « .tpl » représentent une page HTML contenant la carte interactive ainsi que les extensions.

Pour ajouter votre extension à la carte de la PGF :

1. Ajouter vos fichiers « .js » et « .css » dans un dossier sous « **src/content/samples/extensions** ».
2. Ajouter vos références de fichiers dans le fichier « .tpl ».

```
<link rel="stylesheet" href="./extensions/{repertoire}/{fichier}.css" />
<script src="./extensions/{repertoire}/{fichier}.js"></script>
```

3. Ajouter vos **noms d'extensions** à l'attribut « rz-extensions ».

```
<div ... is="rv-map"
  rz-extensions="extension1,extension2, ..."
  rv-config="config.json">
</div>
```

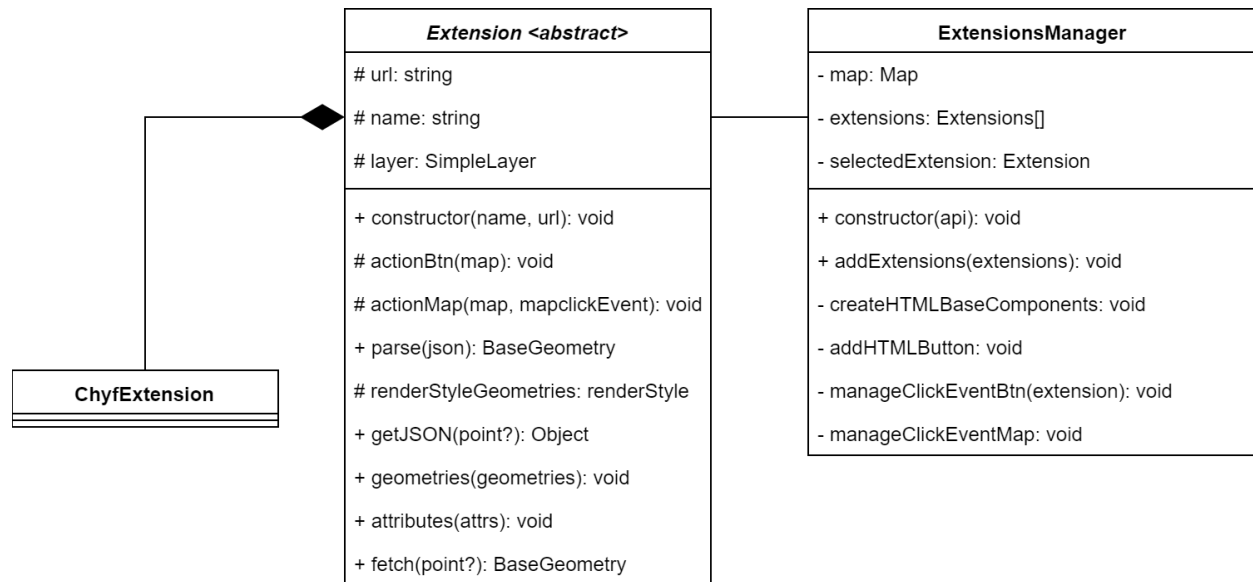
4. Lancer le serveur :

```
npm run serve
```

5. Vous devriez apercevoir vos extensions sur la carte lorsque vous accéder à l'adresse :

```
http://localhost:6001/samples/{nomFichier}.html
```

## Structure des extensions



### Classe Extension

Représente la classe décrivant une extension. Toutes extensions doivent dériver cette classe.

Attributs	Description
<b>url</b>	Chemin de la requête HTTP du JSON (voir méthode <code>getJSON()</code> pour les attributs).
<b>name</b>	Le nom de l'extension.
<b>layer</b>	Le calque sur la carte contenant les éléments graphiques.

Méthode	Description	Élément retourné
<b>constructor(name, url)</b>	Vous devez fournir le nom et le chemin de la requête HTTP du JSON.	
<b>actionMap(map, mapClickEvent)</b>	Événements exécutés lors d'un clique sur la carte.	
<b>actionBtn(map)</b>	Événements exécutés lors d'un clique sur le bouton de l'extension.	
<b>parse(json)</b>	Permet de définir les méthodes de transformation des attributs du JSON en formes géométriques.	Formes géométriques.
<b>renderStyleGeometries</b>	Permet de définir les propriétés graphiques des formes géométriques.	Propriétés graphiques des formes géométriques.
<b>getJSON(point?)</b>	Permet de définir le chemin de la requête HTTP.	JSON.
<b>geometries(geometries)</b>	Remplace les formes géométriques présentes sur le calque.	

<b>attributes(attrs)</b>	Remplace les attributs du calque.	
<b>fetch(point?)</b>	Récupère le JSON « getJSON ». Modifie les attributs « setAttributes ». Conversion du JSON « parse ».	Forme géométriques.

## Classe ChyExtension

Représente la classe décrivant les extensions de CHyF. La classe doit dériver la classe **Extension** pour définir son propre comportement.

## Classe ManageExtension

Représente la classe qui gère l'ensemble du groupe d'extensions. Elle permet, entre autres, de créer les boutons HTML, de créer des couches de configuration, de gérer les événements sur les boutons et de gérer les événements de la carte reliés aux extensions du groupe.

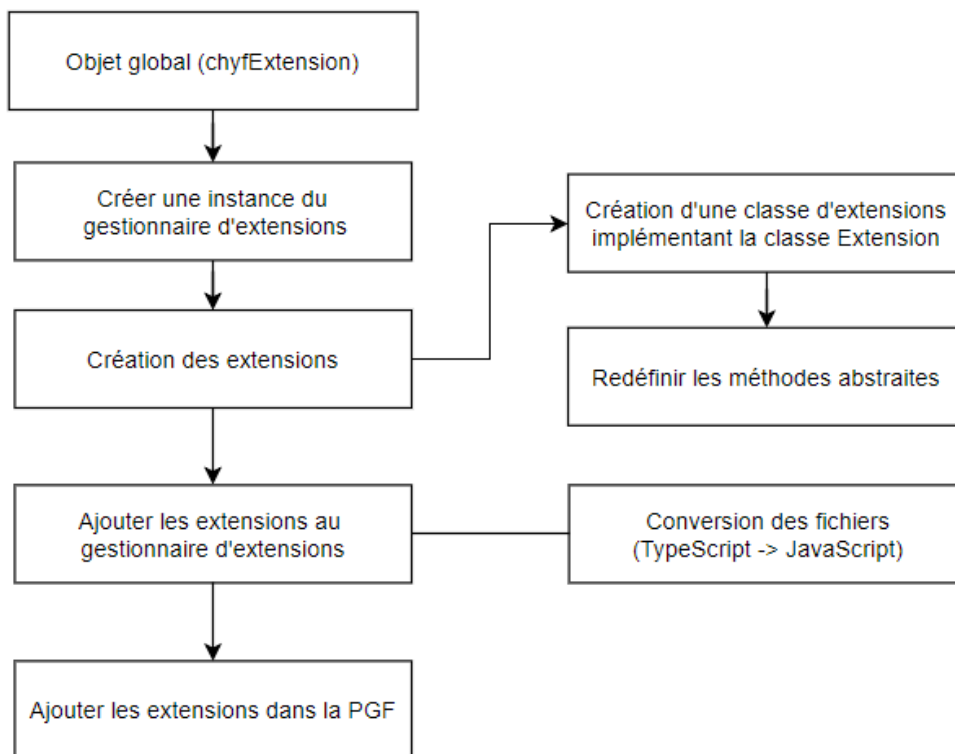
Attributs	Description
<b>map</b>	Instance de la carte de la PGF.
<b>extensions</b>	Liste des extensions.
<b>selectedExtension</b>	L'extension sélectionnée.
<b>extensionsBaseElement</b>	Instance HTML du groupe d'extensions.

Méthode	Description	Élément retourné
<b>constructor(api)</b>	Permet d'initialiser les événements et récupère l'instance de la carte.	
<b>addExtensions(extensions)</b>	Permet d'ajouter des extensions.	
<b>createHTMLBaseComponent</b>	Permet de créer la structure HTML de base des extensions.	
<b>manageClickEventBtn (extension)</b>	Permet de gérer les événements « click » sur une extension. Lors de la sélection d'une extension, on conserve celle-ci.	
<b>deselectAll</b>	Supprime l'état et le style pour chaque bouton sélectionné.	
<b>addHTMLButton(name)</b>	Ajoute un bouton au composant HTML de base des extensions.	
<b>manageClickEventMap</b>	Permet de gérer les événements « click » sur la carte.	

## Flux de travail

Voici l'exécution de la structure des extensions :





Exemple de la structure de l'objet global:

```
(<any>window).myExtension = {
  init: function(api: any) {
    const manageExtension: ManageExtension =
ManageExtension.getInstance(api);
    const upstream: Extension = new CHyFExtension("upstream",{url});
    const downstream: Extension = new
CHyFExtension("downstream",{url});
    manageExtension.addExtensions([upstream, downstream]);
  }
};
```