

Collège Ahuntsic
Département d'informatique

420-289-AH
Programmation web côté
serveur
hiver 2022

TP1

Développement d'une application météorologique

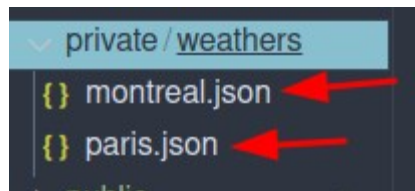
Énoncé

Vous êtes engagé par la firme MétéoWiki pour développer une application permettant d'accéder aux conditions météorologiques de différentes villes. Cette application sera rendue sur le serveur (server side rendering) à l'aide de Handlebars et de Node.js.

Travail à faire

Étape 1: Création des templates [11 points]

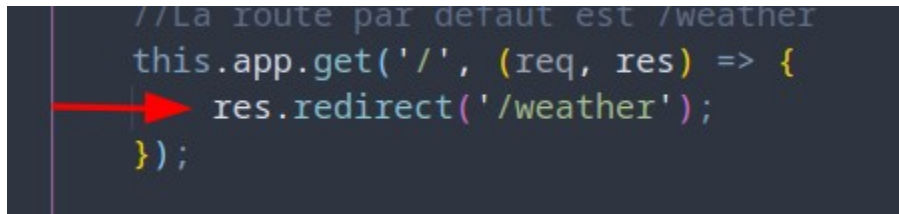
La première étape de la création du site web consiste à créer les templates .hbs pour votre application. Dans le gabarit fourni par MétéoWiki, Express est configuré pour utiliser le dossier *templates/views* pour les vues et *templates/partials* pour les partiels. Pour vous simplifier la vie, le service *src/services/staticWeather.service.ts* a été créé. Ce service permet de fournir à votre serveur les informations météorologiques se trouvant dans le dossier *private*. Ces informations seront utilisées initialement pour simplifier le développement des vues. Les villes *montreal* et *paris* sont disponibles.



Un exemple d'utilisation du service est disponible dans le *src/controllers/weather.controller.ts*. Les services permettent d'éviter d'avoir trop de logique dans nos routes. Le rôle du controller consiste à effectuer la gestion d'un sous-ensemble de route et ainsi d'éviter de définir toutes les routes dans un même fichier .ts. Le *src/controllers/weather.controller.ts* reçoit toutes les routes débutant par */weather*. La variable *wttrInfo* contient les informations de la ville par défaut (Montréal). Vous pouvez donc passer ces informations à vos vues.

```
/*
 * GET /weather/
 * http://localhost:8000/weather/
 */
router.get('/', async (req: Request, res: Response) => {
  //la variable wttrInfo contient la météo de Montréal (ville par défaut)
  const wttrInfo: WttrObject = await this._weatherService.readWeather(this._defaultLocation);
  res.render('index', {data: wttrInfo});
});
```

Dans `src/app.ts` la route racine (/) est redirigée vers la route `weather`.



```
//La route par défaut est /weather
this.app.get('/', (req, res) => {
  res.redirect('/weather');
});
```

Le dossier public est déjà configuré comme un dossier statique.

```
//Configuration du dossier public
this.app.use(express.static(path.join(__dirname, '../public')));
```

Les onglets/routes suivant(e)s devront être implémentés:

- **Conditions actuelles (GET /weather/now):** Onglet résumant les conditions actuelles d'une ou de plusieurs villes (voir le champ `current_condition` de `WttrObject`).
- **Prochaines heures (GET /weather/hourly):** Onglet résumant les conditions pour les prochaines. Chaque champ `weather` contient un champ `hourly`. Ce champ `hourly` indique les prévisions à une certaine heure. L'heure est disponible sous le champ `time` qui représente une certaine heure sous le format [militaire](#). Vous pouvez modifier l'affichage de l'heure à l'aide d'un [helper Handlebars](#).

On désire également pouvoir visualiser les informations de chaque onglet pour plusieurs villes en même temps dans le but de comparer plus facilement les informations. Pour cela, la barre de recherche permettant d'entrer le nom de la ville doit pouvoir supporter **plusieurs villes séparées par des virgules**. Assurez-vous également que la dernière recherche de l'utilisateur soit conservée entre les onglets. Vous pouvez passer l'information en lien avec la recherche dans le url (`localhost:8000/weather?recherche=montreal,paris`).

Un exemple d'interface est disponible en annexe.

Étape 2: Connexion à l'API de `wtr.in` [4 points]

La deuxième étape consiste à faire le lien avec l'API de `wtr.in`. Cette communication doit être faite dans le service `src/services/wtrWeather.service.ts`. Ce service implémente la même interface que le service `src/services/staticWeather.service.ts`, il pourra donc le remplacer facilement une fois fonctionnel.

```
async readWeather(location: string): Promise<WttrObject> {
  throw new Error('Vous devez compléter le wtrWeatherService.ts');
  //TODO Faire une requête GET vers l'api de wtr.in pour obtenir la météo de <location>
  //Vous pouvez utiliser node-fetch pour faire la requête.
}
```

Une fois complété, il suffit de modifier le binding dans le fichier `src/inversify.config.ts` (voir le TODO).

```
// TODO Une fois le wttrService complété, utiliser wttr comme provider au lieu des fichiers statiques.  
// Il vous suffit de d'interchanger les deux import suivants  
import { StaticWeatherService as weatherProvider } from '../services/staticWeather.service';  
import { wttrWeatherService as weatherProvider } from '../services/wttrWeather.service';
```

Le module `inversify` permet l'injection des dépendances dans notre application ([inversify](#)). La documentation sur l'API de `wttr.in` est disponible sur leur [Github](#). Voici des exemples d'utilisation:

wttr.in/montreal?format=j1	Retourne la météo sous format .json pour la ville de Montréal.
wttr.in/@github.com?format=j1	Retourne la météo sous format .json pour la position de l'adresse ip associée à github.com.
wttr.in/~kilimanjaro?format=j1	Retourne la météo sous format .json d'une position près du mont Kilimanjaro.

Si vous utilisez bien l'API, votre application devrait être en mesure d'offrir la météo à partir d'un nom de domaine (p.ex. `github.com`) ou d'un lieu (p.ex. `kilimanjaro`).

Étape 3: Intégration de MongoDB [optionnelle]

Cette partie ne sera pas corrigée, mais si vous désirez aller plus loin cette étape consiste à utiliser MongoDB pour augmenter les performances de votre application en conservant le résultat des requêtes faites par l'utilisateur. Vous devez donc:

- Sauvegarder le résultat de chaque requête faite par un utilisateur dans MongoDB.
- Lorsque l'utilisateur fait une requête présente dans MongoDB, vérifiez que la date et le temps de l'observation sont toujours pertinents (on ne veut pas retourner des données qui ne sont plus d'actualité). Si les données sont assez récentes, les retourner, sinon assurez-vous de les mettre à jour.

La complexité de MongoDB devra être encapsulée dans le service `src/services/mongodb.service.ts`. Utilisez une base de données nommée **tp1** et une collection nommée **weathers**.

Évaluation

La grille d'évaluation pour la partie #1 et partie #2 est disponible à la page suivante. Assurez-vous d'accorder de l'importance à la qualité de votre code avant d'effectuer votre remise:

- Ne pas laisser du code commenté qui ne sert à rien
- Ajouter des commentaires dans les sections complexes/critiques pour documenter votre code
- Ne pas laisser du code mort (code qui ne peut pas être exécuté comme du code à la suite d'un return)
- Éviter la duplication de code.

Remise

La remise de votre code se fera sur LÉA sous le format .zip avant le 13 mars 23h59. **Assurez-vous de ne pas inclure les dossiers `node_modules` au .zip.** Si le temps le permet, une présentation des projets aura lieu en classe. **Le TP1 se fait seul.**

Grille d'évaluation

Étape #1 [11 points]
1. Vue conditions actuelles [1 point]
2. Vue prochaines heures [2 point]
3. Route conditions actuelles [1 point]
4. Route prochaines heures [1 point]
5. Recherche (une ville) [1 point]
6. Recherche (plusieurs villes) [1 point]
7. Consistance entre onglets [1 point]
8. Qualité de l'interface et expérience [1 point]
9. Qualité du code [2 point]

Étape #2**[4 points]**

1. Service fonctionnel

[2 points]

2. Configuration inversify

[1 point]

3. Qualité du code

[1 points]

Annexe



MétéoWiki

Condition actuelle

Prochaines heures

Paris, Ile-de-France, France

Température: 2°C (ressentie -2°C)

Conditions: Clear

Vents: 13Km/h WSW

Visibilité: 10 km

UV: 1

Humidité: 87%

Précipitations: 0.0 mm

Dernière observation: 2022-02-25 03:34 AM
(heure locale)

Montreal, Quebec, Canada

Température: -12°C (ressentie
-15°C)

Conditions: Partly cloudy

Vents: 7Km/h S

Visibilité: 14 km

UV: 1

Humidité: 61%

Précipitations: 0.0 mm

Dernière observation: 2022-02-24 09:48 PM
(heure locale)