Le middleware static

Les `middleware` maintenus par `Express`.

L'équipe d'`Express` maintient un bon nombre de `middleware` :

- `body-parser` : permet de parser le `body` de la requête.
- `compression` : permet de compresser les objets réponses.
- `connect-rid` : permet d'attribuer un identifiant unique à chaque requête.
- `cookie-parser` : permet de parser les cookies.
- `cookie-session` : permet de sauvegarder une session utilisateur dans un cookie côté client.
- `cors` : permet de gérer les requêtes de types `CORS`.
- `csurf` : permet de sécuriser votre API contre les attaques CSRF, nous y reviendrons.
- `errorhandler` : permet de gérer les erreurs en développement.
- `method-override` : permet de modifier les méthodes des requêtes côté serveur (vous n'en aurez a priori jamais besoin).
- `morgan` : un logger paramétrable pour les requêtes `Http`.
- `multer` : permet de gérer l'upload de fichiers.
- `response-time` : permet d'obtenir les temps entre une requête et la réponse du serveur.
- `serve-favicon` : permet de servir un favicon de manière optimale.
- `serve-index` : permet de fournir une page pour naviguer dans les répertoire et les fichiers.
- `serve-static` : permet de servir facilement des fichiers depuis un dossier.
- `session` : un autre middleware pour les sessions basées sur les cookies mais cette fois-ci côté serveur principalement, nous y reviendrons.
- `timeout` : permet de timetout des requêtes après un temps donné.
- `vhost` : utilisé pour les hôtes virtuels. Permettant d'utiliser plusieurs noms de domaines sur le même serveur.

Nous allons étudier en profondeur 4 `middleware` dans ce chapitre, puis en verrons de nombreux autres dans des chapitres ultérieurs.

Le `middleware serve-static`

Ce module est accessible depuis `express.static()`. Il fait partie **des quatre `middleware` qui sont accessibles directement depuis l'objet `express`**.

Ce module s'utilise de la manière suivante :

```
app.use(express.static(path.join(__dirname, 'public')));
```

Rappelez-vous que `__dirname` est mis à disposition par `require` lorsque nous importons le fichier, considéré par `Node.js` comme un module.

Nous sommes certain de ne pas avoir de problème de `path` quel que soit l'environnement en utilisant `path.join` et `__dirname`.

Il faut bien sûr que le `path` vers le dossier soit en relatif par rapport au fichier contenant le `middleware serve-static`.

Si par exemple le dossier `public` est deux niveaux au dessus du fichier contenant le `middleware` il faudra écrire :

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

`serve-static` a de nombreuses options, nous allons étudier les principales :

Voici les principales :

- `cacheControl` : permet de gérer le `header Cache-Control` permettant d'utiliser les options `maxAge` et `immutable`. Par défaut le `header` est mis automatiquement.
- `etag` : permet de gérer le `header ETag` permettant de générer l'identifiant de la version de la ressource. Par défaut le `header` est mis automatiquement.
- `extensions` : permet de définir un tableau d'extensions par défaut. Si le fichier n'est pas trouvé, le `middleware` cherchera avec toutes les extensions passées : par exemple `['html', 'pdf']` .
- `immutable` : permet d'empêcher les clients de faire une requête pour vérifier si le fichier a changé. Permet un `caching` maximum, il est nécessaire de définir l'option `maxAge` dans ce cas. Cette option est à `false` par défaut.
- `index` : par défaut, une requête sur le dossier renverra le fichier `index.html`. Cette option est à `true` par défaut. Vous pouvez la mettre à `false` ou passer un autre fichier par défaut ou un tableau de fichiers par défaut par ordre de préférence.
- `maxAge` : permet de définir un âge maximum pour les fichiers en millisecondes pour la mise en cache. Cette option est à `0` par défaut.
- `setHeaders` : fonction permettant de modifier les `headers`, vous avez à votre disposition les objets `(res, path, stat)`, respectivement l'objet `response`, le `path` vers le fichier renvoyé et `fs.stat`.

Voici un exemple avec deux options :

```
app.use(express.static(path.join(__dirname, 'public'), {
  maxAge: '1d',
  index: 'pastrouve.html',
}))
```