

NodeJS : Express

- Introduction à Express
- MVC
- Installation et utilisation d'Express
- Routes simples
- Routes dynamiques
- Erreur 404
- **Templates**
- Middlewares

Templates

- Au lieu d'écrire directement en JavaScript le code HTML à renvoyer, on utilise des **templates** dans des **fichiers séparés**
- L'application JavaScript envoie au template les variables nécessaires pour construire les pages
- Il existe plusieurs **moteurs de template** et en général, ils gèrent les variables, les conditions, les boucles,...

Liste de moteurs de template

- EJS (Embedded JS)
- Pug (ou Jade)
- Handlebars
- Mustache
- Underscore
- Template7
- Vue.js
- React
- Twigjs
- jTemplates
- doT
- Nunjunks
- Et bien d'autres...

Moteur de template : EJS

- La gestion du HTML est déléguée au moteur de template **EJS** (Embedded JS - <https://ejs.co/>)
- Pour installer EJS => **npm install --save ejs**
- On crée un **fichier d'extension .ejs** dans un sous-dossier « **views** »
- Une variable est utilisée et sa valeur est affichée dans une balise **<%= nomvariable %>**
- On peut utiliser plusieurs paramètres avec des tableaux, des boucles et des conditions

Suite du 1^{er} projet

- Pour **modifier le projet** et **ajouter le moteur de template**, on peut reproduire la procédure de création du projet précédent
 - Utiliser **npm init** (avec les données du projet)
 - Utiliser **npm install --save express**
 - **Installer le nouveau module : npm install --save ejs**
 - Modifier les fichiers du projet (index.js,...)
- Mais comme le fichier **package.json** est déjà créé, on peut l'adapter et utiliser **npm install** (sans options) ce qui va réinstaller tout ce qui était déjà défini et seulement ensuite installer le module EJS et modifier le projet

Utilisation de EJS

- Quand on installe EJS via npm, le module et ses dépendances sont ajoutées dans **package.json**
- Les fichiers de template doivent avoir une extension **.ejs**
- Ces fichiers sont recherchés dans un dossier à la racine du projet **./views** par défaut
- Ces fichiers contiennent du **HTML5**
- On peut **diviser le contenu** d'une page en plusieurs parties pour réutiliser du code commun

Comment afficher une page

- On utilise la méthode **render()** de l'objet **Response** de Express ([voir doc](#))
- Il n'est pas nécessaire d'indiquer l'extension, Express s'en charge selon le moteur installé
 - Pour une page statique
`reponse.render('accueil.ejs');`
 - Pour passer la valeur d'une variable (test)
`var test='Bonjour';`
`reponse.render('mapage.ejs', {msg: test});`
 - Et dans le fichier *mapage.ejs*
`<h1>Le message est <%= msg %></h1>`

Choix du répertoire des vues

- Par défaut, le **répertoire des vues** est placé à la racine du projet dans un dossier s'appellant **views**
- Dans Express, **si on souhaite modifier** le nom ou l'emplacement de ce répertoire, on doit :
 - Ajouter le module **path**
const path = require('path');
 - Définir le nouveau chemin (/vues) par rapport à la racine du projet (qui est défini dans **__dirname**)
app.set('views', path.join(__dirname, './vues'));

Template EJS (exemple)

```
const express = require('express');
const app = express();

app.get('/', function(requete, reponse){
  reponse.render('accueil.ejs');
})
.get('/about', function(requete, reponse){
  reponse.render('apropos.ejs');
})
.get('/page/numero=:pagenum', function(requete, reponse){
  const monParametre = requete.params.pagenum;
  // exemple : on accepte uniquement des nombres de 1 à 5
  if (Number.isInteger(Number(monParametre)) && ((monParametre > 0) && (monParametre <= 5))){
    reponse.render('mapage.ejs', {num: monParametre});
  } else {
    reponse.status(404).render('erreur404.ejs');
  }
})
.use(function(requete, reponse, next){
  reponse.status(404).render('erreur404.ejs');
});

app.listen(8080); // port 8080
console.log("Express démarré");
```

Les pages (template EJS)

accueil.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Accueil</title>
    <meta charset='utf-8' />
  </head>
  <body>
    <h1>Vous êtes sur la page d'accueil !</h1>
    <p>Bienvenue sur notre site.</p>
  </body>
</html>
```

apropos.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>A propos</title>
    <meta charset='utf-8' />
  </head>
  <body>
    <h1>Vous êtes sur la page A propos (about)</h1>
    <p>Et vous lisez le fichier apropos.ejs</p>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page n°<%= num %></title>
    <meta charset='utf-8' />
  </head>
  <body>
    <h1>Vous êtes sur une page dynamique !</h1>
    <p>Vous lisez le fichier mapage.ejs</p>
    <strong>Le paramètre passé dans l'URL vaut : <%= num %></strong>
  </body>
</html>
```

mapage.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Erreur 404</title>
    <meta charset='utf-8' />
  </head>
  <body>
    <h1>Attention, erreur 404 !</h1>
    <p>La page que vous demandez n'existe pas, ou alors le paramètre n'est pas valide.</p>
  </body>
</html>
```

erreur404.ejs

Sous-template

- On peut diviser le HTML en plusieurs portions (par exemple : header, menu, footer,...) et ensuite les regrouper pour construire les pages
 - On utilise la notation
`<%- include('portion'); -%>`
où **portion** est le fichier EJS (portion.ejs) contenant la portion de code à inclure à l'intérieur d'un autre fichier EJS

Exemple (sous-template)

menu.ejs

```
<nav>
  <a href=".">Accueil</a>
  <a href="./about">A propos</a>
  <a href="./presentation">Présentation</a>
  <a href="./page/numero=1">Page 1</a>
</nav>
```

header.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Présentation</title>
    <meta charset='utf-8' />
  </head>
  <body>
    <%- include('menu'); -%>
    <header>
      <h1>Présentation</h1>
    </header>
```

```
    <footer>Copyright CF2M - &copy; 2021</footer>
  </body>
</html>
```

footer.ejs

```
<%- include('header'); -%>
<main>
  <h2>Contenu principal de la page Présentation</h2>
  <p>Placé entre le menu, l'entête et le pied de page.</p>
  <p>Et vous lisez le fichier presentation.ejs</p>
</main>
<%- include('footer'); -%>
```

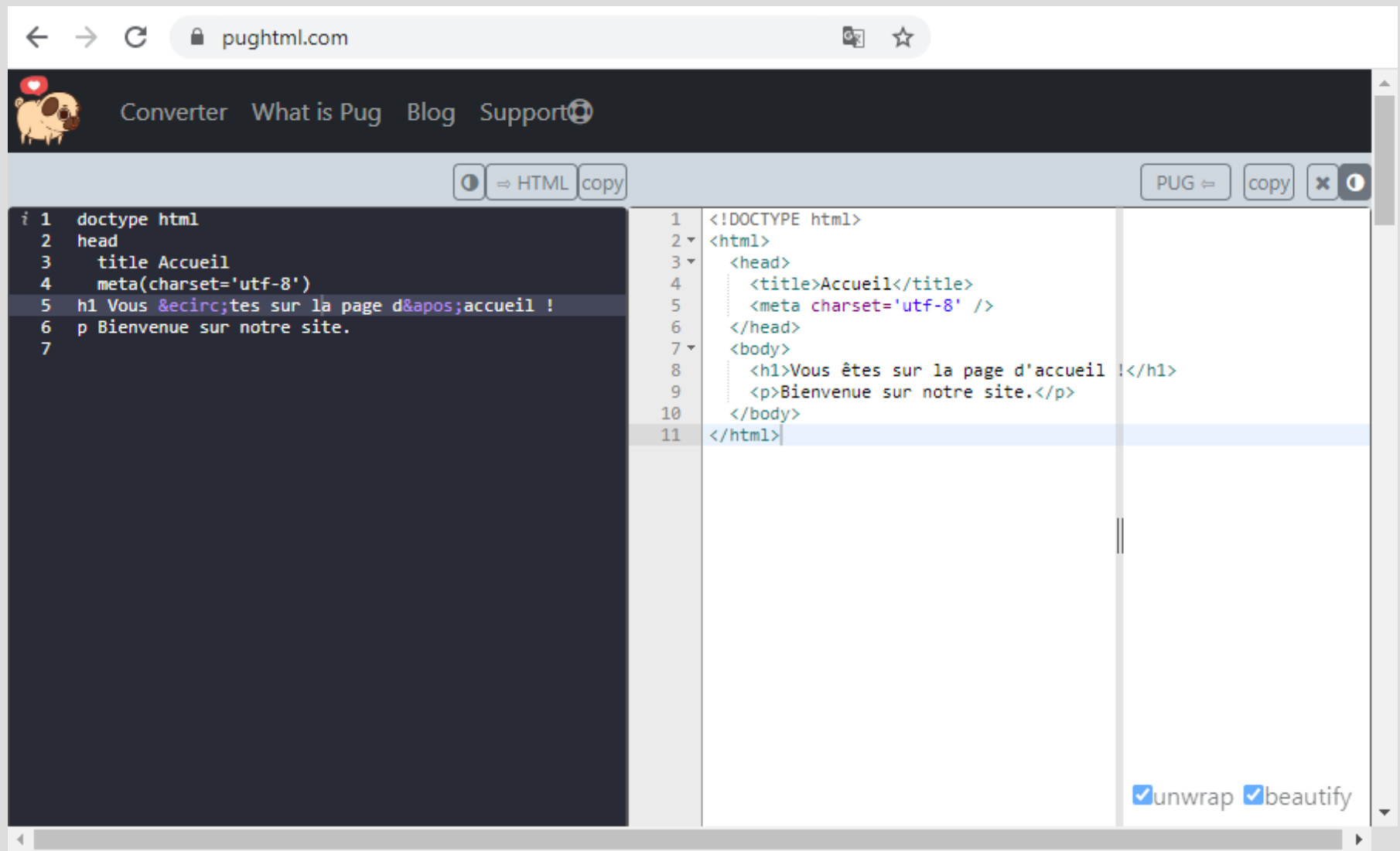
presentation.ejs

```
.get('/presentation', function(requete, reponse, next){
  reponse.render('presentation.ejs');
})
```

Choix du moteur de template

- Dans Express, si on souhaite **utiliser un autre moteur** de template que EJS, il faut :
 - **Installer ce module avec npm**
exemple : ***npm install --save pug***
 - **Indiquer à Express quel moteur utiliser** en le déclarant dans le code (index.js)
exemple : ***app.set('view engine', 'pug');***
- NB : si un seul moteur est chargé, ce qui est le cas le plus courant, il ne semble pas nécessaire d'utiliser l'instruction `app.set()` car il est configuré d'office par **Express**

Exemple de code avec PUG



The screenshot shows the pughtml.com website interface. The top navigation bar includes a pug dog logo, a heart icon, and links for 'Converter', 'What is Pug', 'Blog', and 'Support'. Below the navigation bar, there are two tabs: '⇒ HTML' and 'PUG ⇐'. The '⇒ HTML' tab is active, showing the converted HTML code. The 'PUG ⇐' tab is also visible, showing the original Pug code. The Pug code is as follows:

```
1 doctype html
2 head
3   title Accueil
4   meta(charset='utf-8')
5   h1 Vous êtes sur la page d'accueil !
6   p Bienvenue sur notre site.
7
```

The converted HTML code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Accueil</title>
5     <meta charset='utf-8' />
6   </head>
7   <body>
8     <h1>Vous êtes sur la page d'accueil !</h1>
9     <p>Bienvenue sur notre site.</p>
10  </body>
11 </html>
```

At the bottom right of the interface, there are two checkboxes: 'unwrap' and 'beautify', both of which are checked.