



TP : Serveur NodeJS 3

1. git

Branch templating

Pousser la version fonctionnelle du templating sur une branche.

Branch object

Créer une nouvelle branche en partant de la branche « templating » précédente.

PR

Créer une Pull Request de la branche « templating » et accepter cette PR (Github).

Fetch

Mettre à jour son repository local avec les changements effectué sur Github.

Rebases

Nous allons repositionner (commande rebase) toutes nos branches par rapport à l'acceptation de la PR qui vient d'être faite sur Github.

Repositionner la branche « master » en local.

Repositionner la branche « object » par rapport à master.

2. Introduction des Promises

Afin de rendre possible l'utilisation de la méthode « .then() » au lieu de passer une méthode de callback au moment de l'appel. Nous allons implémenter cette méthode « query() » dans la Classe DbManager.

```
query(sql, args) {  
    return new Promise( ( resolve, reject ) => {  
        this._db.query( sql, args, ( err, rows ) => {  
            if ( err )  
                return reject( err );  
            resolve( rows );  
        } );  
    } );  
}
```

Nous allons grâce à cela, pouvoir exporter notre

3. Structuration de code

Nous allons créer des Classes pour intégrer les requêtes SQL et structurer notre accès aux données.

ProductRepository

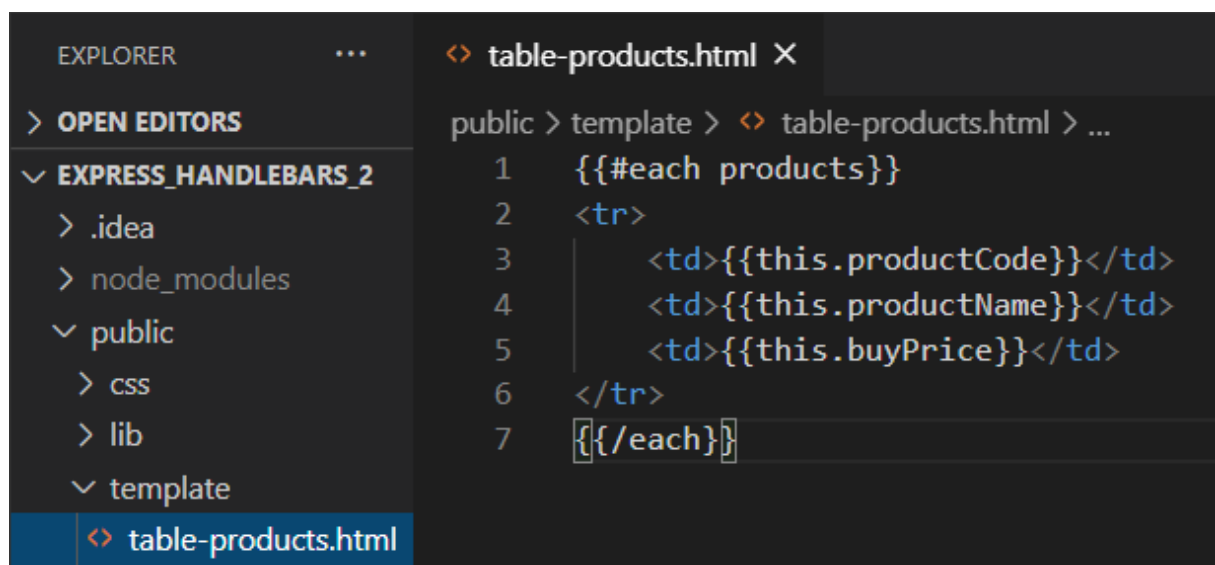
Le constructeur doit prendre en paramètre le dbManager et le stocker dans un de ses attributs.

Migrer le « code SQL » de votre fichier « app.js » dans les deux méthodes à implémenter :

- findAll()
- searchByName(text)

4. Charger dynamiquement un template (optionnel)

Pour éviter le conflit de templating côté backend, nous avons aussi la possibilité d'exposer le template dans les fichiers statiques :



```
EXPLORER  ...  
> OPEN EDITORS  
✓ EXPRESS_HANDLEBARS_2  
  > .idea  
  > node_modules  
  ✓ public  
    > css  
    > lib  
    ✓ template  
      <> table-products.html  
  
<> table-products.html X  
public > template > <> table-products.html > ...  
1  {{#each products}}  
2  <tr>  
3      <td>{{this.productCode}}</td>  
4      <td>{{this.productName}}</td>  
5      <td>{{this.buyPrice}}</td>  
6  </tr>  
7  {{/each}}
```

Charger dynamiquement le template avec jQuery

```
let templateProductRows;

function loadProductsTemplate() {
  $.get('/template/table-products.html', (html) => {
    templateProductRows = Handlebars.compile(html);
  });
}
```

Utiliser ce template déjà compilé pour insérer le HTML dans la page (index.js)

```
function searchData() {
  const search = $inputSearch.val();

  console.log({search});

  if(search) {
    $.get('/search?search=' + search, (products) => {
      $tableBody.html(templateProductRows({ products }));
    });
  } else {
    document.location = '';
  }
}
```

Nettoyer home.hbs

On peut maintenant supprimer ce template du fichier home.hbs