self-service-machine-api - STEP 9

Dans cette étape nous allons terminer la gestion des statuts HTTP.

Amélioration du code pour la route PUT /api/products/:id

Dans notre code, nous avons du code dupliqué.

En effet, nous avons 2 fois le code :

```
...
.catch((error) => {
      const message =
        "Le produit n'a pas pu être mis à jour. Merci de réessayer dans quelques instants.";
      res.status(500).json({ message, data: error });
    });
...
```

Nous allons corriger cela en ne "catchant" une éventuelle erreur que dans le `catch` le plus englobant.

Pour faire cela, il faut bien comprendre que nous devons ajouter un `return` devant Product.findByPk(). Ainsi en cas d'erreur dans le corps de la méthode Product.findByPk() une erreur sera levée et comme `Product.findByPk()` retourne une promesse, elle sera transmise au bloc parent.

Voici le code corrigé :

```
...
productsRouter.put("/:id", (req, res) => {
  const productId = req.params.id;
  Product.update(req.body, { where: { id: productId } })
    .then((_) => {
      return Product.findByPk(productId).then((updatedProduct) => {
        if (updatedProduct === null) {
          const message =
            "Le produit demandé n'existe pas. Merci de réessayer avec un autre identifiant.";
          // A noter ici le return pour interrompre l'exécution du code
          return res.status(404).json({ message });
        }
        // Définir un message pour l'utilisateur de l'API REST
        const message = `Le produit ${updatedProduct.name} dont l'id vaut ${updatedProduct.id} a été mis à jour avec su

        // Retourner la réponse HTTP en json avec le msg et le produit créé
        res.json(success(message, updatedProduct));
      });
    })
    .catch((error) => {
      const message =
        "Le produit n'a pas pu être mis à jour. Merci de réessayer dans quelques instants.";
      res.status(500).json({ message, data: error });
    });
});
...
```

La route DELETE /api/products/:id

Rien de nouveau dans cette section.

Ce sera simplement um bon exercice pour voir si vous avez compris ce que nous avons fait précédemment.

```
...
productsRouter.delete("/:id", (req, res) => {
  Product.findByPk(req.params.id)
    .then((deletedProduct) => {
      if (deletedProduct === null) {
        const message =
          "Le produit demandé n'existe pas. Merci de réessayer avec un autre identifiant.";
        // A noter ici le return pour interrompre l'exécution du code
        return res.status(404).json({ message });
      }
      return Product.destroy({
        where: { id: deletedProduct.id },
      }).then((_) => {
        // Définir un message pour le consommateur de l'API REST
        const message = `Le produit ${deletedProduct.name} a bien été supprimé !`;

        // Retourner la réponse HTTP en json avec le msg et le produit créé
        res.json(success(message, deletedProduct));
      });
    })
    .catch((error) => {
      const message =
        "Le produit n'a pas pu être supprimé. Merci de réessayer dans quelques instants.";
      res.status(500).json({ message, data: error });
    });
});
...
```

Passons à l'étape n°10