

Open-beer Database

Institut Universitaire de Technologie

Département Informatique - Campus III Ifs

Module Clients riches web

-- Ressources initiales

- Faire un Fork du [projet GitHub initial](#)
- Consulter en ligne le projet Exemple : open-beer.kobject.net
- Télécharger et installer le serveur REST : <https://github.com/jcheron/rest-open-beer>

ou

- utiliser l'api en ligne <http://rest-open-beer.kobject.net>

-- Présentation générale du projet

L'application fournie permet de gérer un ensemble de données relatives aux brasseurs et aux bières. Elle accède à une API REST en ligne.

Le projet consiste à reprendre cette application AngularJS existante, en l'améliorant et en y ajoutant les fonctionnalités demandées.

Il est rappelé que **l'abus d'alcool est dangereux pour la santé, à consommer avec modération** et que l'abstinence de toute consommation est recommandée à un certain nombre de personnes, notamment aux mineurs en-dessous de l'âge légal.

-- Existant

-- Fonctionnalités

-- Configuration

HOME / CONFIG

Rafraîchissements

Breweries

Recharger à chaque affichage
 A la demande

Mise à jour

Breweries

Immédiate
 A la demande

Paramètres d'accès au serveur distant

URL

http://127.0.0.1/rest/

Private token

Entrez la clé

Force connexion

SAUVEGARDER LES MODIFICATIONS ANNULER

-- Serveur distant

L'adresse du serveur REST distant est précisée dans le champ **URL**. La connexion avec authentification au server (non mise en place) à partir d'une adresse mail et d'un mot de passe (Stockés dans la table **User**) délivre un "**token**" valable la durée d'une session, permettant l'envoi de requêtes de modifications. En l'absence d'authentification (de manière provisoire), le paramètre **force connexion** permet de faire les mises à jour sans authentification.

-- Mises à jour

L'application peut fonctionner selon 2 modes :

1. **Connecté au serveur** : dans ce cas, les données sont rechargées depuis le serveur à chaque affichage, et les mises à jour sont immédiates
2. **Hors connexion** : les données ne sont chargées que la première fois, les mises à jour sont différées, sauvegardées localement, et exécutées à la demande.

Dans le mode hors connexion, les opérations en attente son consultables, il est possible de les annuler, ou de les exécuter immédiatement :

HOME / SAVES

3 OPERATIONS

<input type="checkbox"/>	Operation ↓↑	#	Objet	Created at	Updated at
<input type="checkbox"/>	Deleted	9	Lagunitas	2015-03-07 02:20:29	2015-03-07 02:20:29
<input type="checkbox"/>	New		Fischer	08/03/15 15:36:30	
<input type="checkbox"/>	Updated	13	Founders2	2015-03-07 02:22:58	08/03/15 14:51:45

-- Liste des brasseurs

La liste des brasseur peut être filtrée ou ordonnée (suivant les champs). Elle est actualisée en temps réel en fonction des options de configuration.

Sélection : Par un click sur la ligne d'une brasserie, il est ensuite possible de cliquer sur le bouton **Modifier** devenu apparent.

HOME / BREWERIES

11 BREWERIES

<input type="checkbox"/>	#	Name ↓↑	url	Created at	Updated at
<input type="checkbox"/>	1	Anchor	http://www.anchorbrewing.com/	2015-03-06 19:11:39	2015-03-06 21:53:12
<input type="checkbox"/>	11	Boston Beer Company	http://www.samadams.com/	2015-03-07 02:21:38	2015-03-07 02:21:38
<input type="checkbox"/>	14	Brooklyn	http://www.brooklynbrewery.com/	2015-03-07 02:23:38	2015-03-07 02:23:38

Sélection pour suppression : Par un click sur de la ou des cases à cocher précédent une ou des brasseries, il est ensuite possible de cliquer sur le bouton **Supprimer** devenu apparent.

HOME / BREWERIES

11 BREWERIES

<input type="checkbox"/>	#	Name ↓↑	url	Created at	Updated at
<input type="checkbox"/>	1	Anchor	http://www.anchorbrewing.com/	2015-03-06 19:11:39	2015-03-06 21:53:12
<input checked="" type="checkbox"/>	11	Boston Beer Company	http://www.samadams.com/	2015-03-07 02:21:38	2015-03-07 02:21:38
<input checked="" type="checkbox"/>	14	Brooklyn	http://www.brooklynbrewery.com/	2015-03-07 02:23:38	2015-03-07 02:23:38
<input type="checkbox"/>	6	Dogfish Head	http://www.dogfish.com/	2015-03-07 02:19:22	2015-03-07 02:19:22

Accès direct à la modification : un double clic sur la ligne d'une brasserie affiche le formulaire de modification

-- Modification de brasseur

HOME / BREWERIES / UPDATE

Nom

Brooklyn

URL

http://www.brooklynbrewery.com/

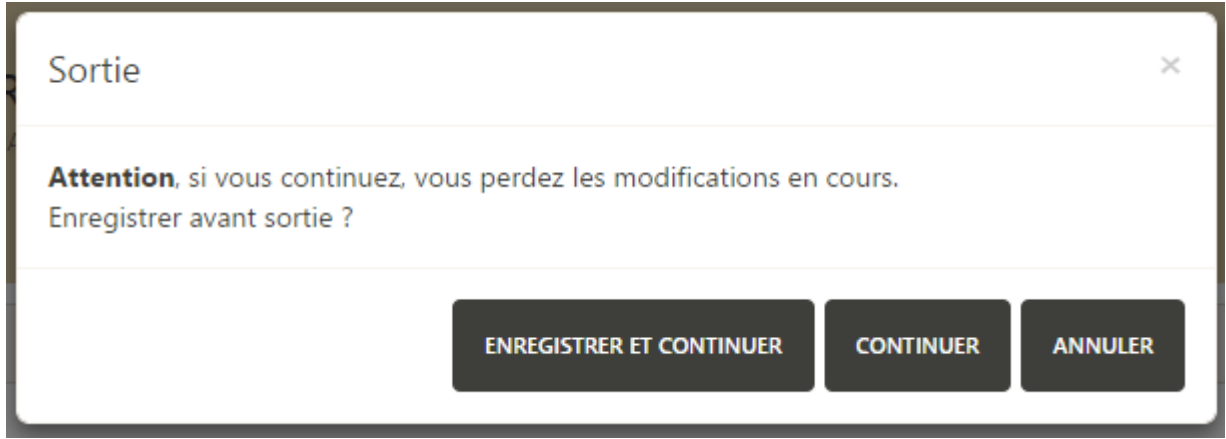
Date de création

2015-03-07 02:23:38

VALIDER

ANNULER

Contrôles de validité côté client : le nom ne doit pas être null, l'url doit être bien formée
Contrôles de validité côté serveur : il est impossible de donner un nouveau nom identique à un nom de brasserie existant (error 409 : CONFLICT)
Contrôles sur sortie : Une modification et tentative de sortie sans validation ou Annulation affiche un message d'avertissement :



En mode hors connexion :

la mise à jour apparaît dans la liste des opérations en attente.

<input type="checkbox"/>	Operation ↓↑	#	Objet	Created at	Updated at
<input type="checkbox"/>	Updated	14	Brooklyn2	2015-03-07 02:23:38	08/03/15 16:56:00

La tentative de modification d'une brasserie n'existant plus (supprimée par un autre utilisateur) provoque une erreur 404 et affiche un message :

i Informations

Erreur de connexion au serveur, statut de la réponse : 404

Mise à jour : La brasserie Brooklyn n'existe plus dans la base de données.

-- Modification de brasseur

L'ajout comporte les mêmes fonctionnalités que la modification.

-- Suppression de brasseur

La suppression nécessite de cliquer sur la case à cocher d'un ou plusieurs brasseurs, puis de cliquer sur le bouton supprimer les brasseries.

En mode connecté, les suppressions sont immédiates et sans message de confirmation.

En mode hors connexion, les suppressions apparaissent dans les opérations en attente (et peuvent donc être annulées) :

HOME / SAVES

Filtrer...

3 OPERATIONS

<input type="checkbox"/>	Operation ↓↑	#	Objet	Created at	Updated at
<input type="checkbox"/>	Deleted	11	Boston Beer Company	2015-03-07 02:21:38	2015-03-07 02:21:38
<input type="checkbox"/>	Deleted	14	Brooklyn2	2015-03-07 02:23:38	2015-03-08 17:00:57

La tentative de suppression d'une brasserie n'existant plus provoque une erreur 404 et affiche un message.

La tentative de suppression d'une brasserie possédant des bières provoque une erreur 409 (CONFLICT) et affiche le message :

i Informations

Erreur de connexion au serveur, statut de la réponse : 409

Impossible de supprimer 'Anchor' dans la base de données.

SQLSTATE[23000]: Integrity constraint violation: 1451 Cannot delete or update a parent row: a foreign key constraint fails ('open-beer', 'beer', CONSTRAINT 'beer_ibfk_1' FOREIGN KEY ('idBrewery') REFERENCES 'brewery' ('id'))

-- Service REST

voir le descriptif : [Service Rest](#)

-- Structure technique de l'application AngularJS

--Schéma général



-- Modules

Application	mainApp (js/app.js)
Dépendances	BreweriesApp (js/breweries/breweriesModule.js) ConfigApp (js/config/configModule.js)

Contrôleurs	MainController (js/mainController.js)	
	hasOperations() Retourne vrai si des opérations sont en attente (mode déconnecté)	
	opCount() Retourne le nombre d'opérations en attente	
	SaveController (js/save/saveController.js)	
	data objet save (opérations en attente)	
	allSelected variable d'état permettant de (dé)sélectionner toutes les opérations	
	sortBy Mémorise le classement des opérations	
	saveAll() Envoie vers le serveur toutes les opérations en attente	
	setActive(operation) Définit l'opération active (non utilisé)	
	isActive(operation) Retourne vrai si l'opération est active	
	countSelect() Retourne le nombre d'opérations sélectionnées	
	remove() Supprime la ou les opérations sélectionnées	
	Module	BreweriesApp (js/breweries/breweriesModule.js)
	Contrôleurs	BreweriesController (js/breweries/breweriesController.js)
data Données à afficher		
sortBy Mémorise le classement des données		
messages Messages provenant du service Rest		
allSelected variable d'état permettant de (dé)sélectionner toutes les lignes		
selectAll() Sélectionne toutes lignes en vue de leur suppression (case à cocher)		
refresh() Uniquement en mode déconnecté, exécute les opérations en attente et rafraichit les données		
showUpdate() Affiche le bouton Modifier si une ligne est active		
refreshOnAsk() Retourne vrai si la config définit que les données ne sont rafraichies qu'à la demande		
defferedUpdate() Retourne vrai si les mises à jour sont différées (mode déconnecté)		
setActive(brewery) (dés)active une ligne		
isActive(brewery) Retourne vrai si la ligne est active		
hasMessage() Retourne vrai si le service Rest a émis 1 ou des messages		
readMessage() Affiche un message Rest pendant une durée définie		
countSelected() Retourne le nombre de lignes sélectionnées (cases à cocher)		
hideDeleted() Affiche/masque les lignes supprimées		
edit(brewery) Affiche le formulaire d'édition de l'objet brewery		
update(brewery, force, callback) Met à jour brewery dans la BDD		
remove()		

Module	ConfigApp (js/config/configModule.js)
Contrôleur	ConfigController (js/config/configController.js)
	config Configuration actuelle
	setFormScope(form) Définit le formulaire actif
	update() Met à jour le formulaire et retourne à l'accueil
	cancel() Annule les modifications et retourne à l'accueil

-- Services

Service	Rest (js/services/rest.js)
	headers Définit en JSON les en-têtes HTTP par défaut
	getParams() Retourne la chaîne passée dans l'url des requêtes (queryString)
	addMessage(message) Ajoute le message à l'ensemble des messages
	clearMessages() Supprime tous les messages en cours
	getAll(response,what) Retourne tous les enregistrement de what dans response
	post(response,what,name,callback) Ajoute une instance de what postée dans response.posted , appelle la fonction de callback si elle est définie
	put(id,response,what,name,callback) Modifie l'instance d'identifiant id par les paramètres postés dans response.posted , appelle la fonction de callback si elle est définie
	remove(object,what,callback) Supprime l'instance object , appelle la fonction de callback si elle est définie
	Save (js/services/save.js)
operations Tableau des opérations en attente	
addOperation(type,operation,object) Ajoute une opération du type (new, update, delete), operation est la fonction à appeler sur object	
execute(index) Exécute l'opération à la position index	
executeAll() Exécute toutes les opérations en attente	

-- Travail à effectuer

-- Modalités

- Travail en Binômes
- sur 2 séances de TD et entre ces 2 séances
- Créer au besoin un compte gitHub (si vous n'en possédez pas)
- L'un des 2 membres doit faire un fork du [projet GitHub initial](#)
- L'autre membre de l'équipe doit cloner le projet de son co-équipier
- Travailler ensuite en continu, en faisant des commits réguliers et commentés, veillez à faire en sorte que

les commits révèlent la régularité et l'intensité du travail de chacun des membres de l'équipe (sans chercher à calculer...)

-- Fonctionnalités à implémenter

- **TODO** Voir [Projet Open-beer - fonctionnalités à implémenter](#)

-- "A rendre"

"A rendre" entre guillemets, puisque votre projet est accessible sur Github :

- Le **fork** de l'application avec fonctionnalités implémentées
- Créer le fichier **Readme.md** pour présenter le travail réalisé :
 - Veillez à y intégrer les noms des membres de l'équipe
 - Son contenu doit permettre d'apprécier l'implémentation des fonctionnalités demandées (ou non demandée et implémentées)
 - d'un point de vue technique (structure, modules, contrôleurs, services...)
 - d'un point de vue fonctionnel (du point de vue de l'utilisateur)
 - Le fichier readme peut faire référence à des liens externes (pages supplémentaires hébergées...), images...
 - Le fichier readme doit mettre en avant le projet, sur le fond (son contenu) et sur la forme (esthétique)

From:
<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:
<http://slamwiki2.kobject.net/slam4/ricclient/angularjs/project/openbeerdatabase?rev=1426293753>

Last update: **2019/08/31 14:40**

