

Présentation générale

Table des matières

Introduction	3
Présentation générale	3
LMS et Studio	4
Installation d'une machine virtuelle OpenFUN / edX	6
Téléchargement (optionnel mais fortement recommandé)	7
Clonage des dépôts de code Open edx et OpenFUN (optionnel mais recommandé aux développeurs)	7
Lancement de la machine virtuelle	7
Lancement d'un serveur web	8
Le forum	9
Open edX et les différents types d'activités	9
Mon premier Xblock 'Hello Student !'	10
Installer le xblock sdk depuis le dépôt Github	10
Lancer le serveur de développement	10
Créons la structure de notre xblock	10
Afficher 'Hello student'	11
Enregistrer notre xblock dans le workbench.	11
Liens utiles	12

JS-Input	12
Introduction	12
Les mécanismes de base	12
Chargement de l'activité et initialisation de l'état	15
Vérification du problème coté Open edX	16
Mécanismes de retour d'information	18
Les modules	18
Custom Response Problem et JS Input Problem	18
JSChannel	18
Trucs et astuces	18
Intégrer du JS Input directement de github	18
Faire une activité qui retourne une note différente de 0 ou 1	19
Liens utiles	19
Analytics	19
Introduction	19
Téléchargement des fichiers de logs	19
Source des logs	19
Format des logs	19
Analyse des logs à l'aide de Elasticsearch	20
Installation	20
Envoi des logs vers Elasticsearch	20
Visualisation des résultats dans Kibana	20
Réaliser des requêtes manuelles sur Elasticsearch	20
A quoi ça ressemble ?	21
Modifier l'apparence d'Open edX	21
Solution 1 : Thème Stanford	21
Pas à pas avec le "Stanford Theme"	22
Thème IonisX	22
Thème FUN	22

Liens Utiles	22
Thème Stanford (utilisé par FUN-MOOC)	22
Exemples de thèmes	23
Documentation générale	23
Photos d'écran et suggestions	23
Ecran d'accueil et écrans externes au cours	23
L'écran d'accueil	23
Le footer	24
Le tableau de bord étudiant et sa liste des cours	24
Pages d'inscription et de connexion	24
S'inscrire	24
Se connecter	24
Le syllabus d'un cours	24
Le syllabus d'un cours	26
Intérieur d'un cours	26
Le forum	26
Le contenu du cours	26
Administration	26
MySQL	26
MongoDB	28
Les logs	28

Introduction

Cette documentation est la version pdf du site disponible sur <http://openfun.github.io/hackathon/> et dans le [dépôt github correspondant](#).

Présentation générale

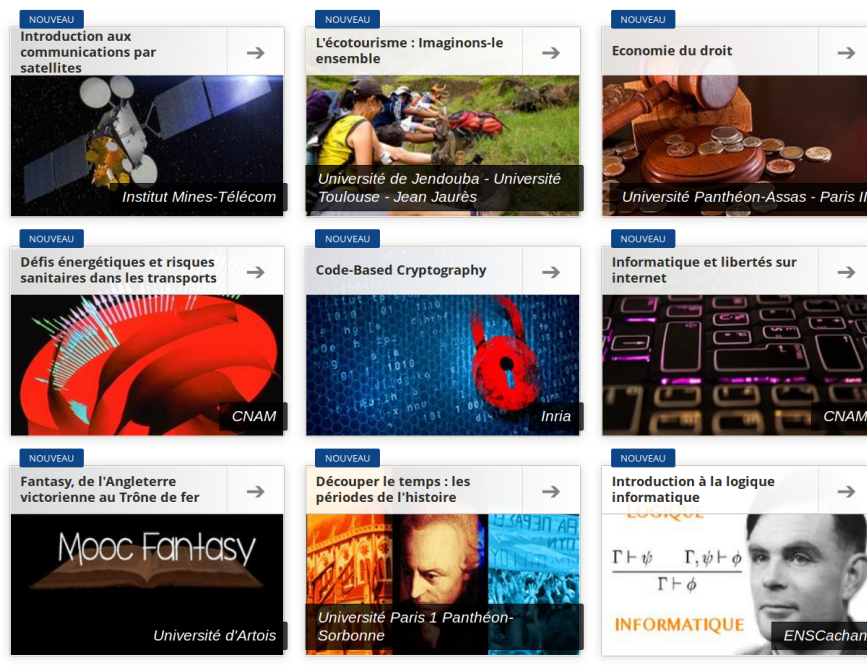
edX est une plateforme web qui permet de délivrer des cours en ligne ouverts à tous, les MOOCs - Massive Online Open Courses.

Les cours sont édités dans le « Studio ». Le Studio est une section réservée aux enseignants et aux personnes responsables de maintenir les cours. Il s'agit d'une interface web d'où l'on peut éditer les contenus de cours, gérer les vidéos et les autres ressources à destination des apprenants. Depuis le Studio, on peut gérer les calendriers de cours, gérer les barèmes de notation des apprenants, concevoir les quiz, etc.

Le LMS (Learning Management System), est la section publique de la plateforme edX. Les cours qui sont édités dans le Studio sont publiés dans le LMS et disponibles aux apprenants. Le LMS est la partie la plus exposée et permet notamment aux apprenants de se connecter et de suivre leur cours. Il s'agit de <https://www.france-universite-numerique-mooc.fr> par exemple. Les apprenants peuvent s'inscrire, consulter les cours, répondre aux quiz, accéder aux résultats, etc.

LMS et Studio

Lorsque vous installez OpenFUN ou Open edX, vous aurez les deux “sites” LMS et Studio sur votre machine de développement.



Les moocs ne sont pas composés uniquement de vidéos de cours. Ils s'accompagnent aussi de nombreuses activités, jeux...

Les étudiants suivent des moocs depuis le LMS (Learning Management System) :

DRAG AND DROP (1 point possible)

Remettez dans l'ordre.

◀	/body	/head	doctype	html	meta	head	▶
---	-------	-------	---------	------	------	------	---



Vérifier

Afficher la réponse

HISTORIQUE DES SOUMISSIONS

INFO DE DÉBOGAGE POUR L'ÉQUIPE PÉDAGOGIQUE

DROPDOWN (4/8 points)

Faire correspondre les éléments suivants :

p

paragraphe ✓

img

image ✓

hr

 ✗

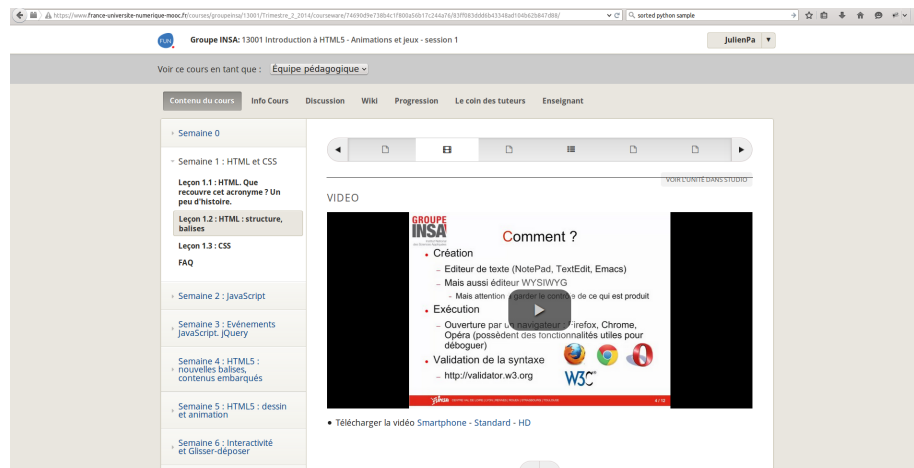
br

 ✓

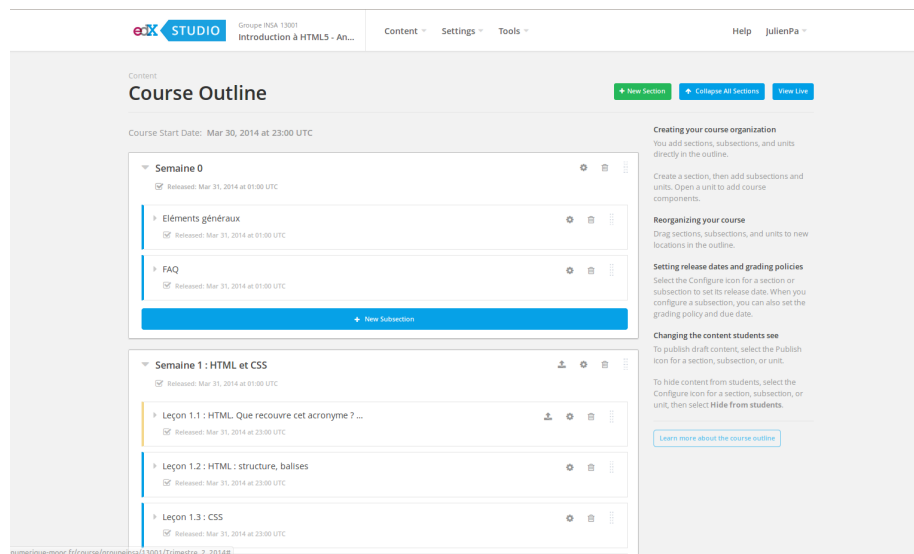
em

gras ✗

etree



Et les professeurs conçoivent ces moocs depuis un *CMS* (Content Management System), appelé aussi *Studio* :



Installation d'une machine virtuelle OpenFUN / edX

Les composants nécessaires à l'installation de FUN ou d'edX sont nombreux et relativement complexes ; c'est pourquoi il existe des machines virtuelles (VM) disponibles en simple téléchargement qui permettent de commencer rapidement

à tester ces applications. Dans la suite de cette section, nous allons voir les étapes à suivre pour obtenir un environnement de développement fonctionnel.

Téléchargement (optionnel mais fortement recommandé)

Les VM OpenFUN sont disponibles au téléchargement via bittorrent. Si vous ne disposez pas d'un client bittorrent (tel que [Transmission](#), [Vuze](#) ou [Deluge](#)), vous devrez télécharger les VM en HTTP, ce qui risque d'être plus lent et de saturer les serveurs de FUN.

Les fichiers .torrent correspondant aux différentes version d'OpenFUN sont disponibles ici : <http://files.alt.openfun.fr/vagrant-images/fun/>

Vous pouvez télécharger le fichier openfun-*.torrent correspondant à la version la plus récente d'OpenFUN dans votre client bittorrent favori.

Avant de créer votre VM, il faudra indiquer à Vagrant le répertoire dans lequel vous avez téléchargé les images :

```
export VAGRANT_BOXES=/chemin/vers/mon/repertoire/de/torrents/  
export FUN_RELEASE=2.11 # Si vous avez téléchargé la version 2.11 d'OpenFUN
```

Clonage des dépôts de code Open edX et OpenFUN (optionnel mais recommandé aux développeurs)

Cette étape optionnelle est néanmoins bien pratique si vous comptez contribuer au code d'Open edX ou de FUN. En effet, vous voudrez vraisemblablement éditer le code source dans votre machine hôte avec votre IDE favori avant de voir le résultat dans votre VM. Pour cela :

```
# Choisissez un répertoire dans lequel cloner les dépôts de code nécessaires  
mkdir /home/user/repos  
cd /home/user/repos/  
  
# Clonez les dépôts  
git clone https://github.com/openfun/fun-apps  
git clone https://github.com/openfun/edx-platform # cela peut prendre un peu de temps...  
mkdir themes && git clone https://github.com/openfun/edx-theme themes/fun/  
  
# Indiquez à Vagrant le répertoire dans lequel vous avez cloné les dépôts  
export VAGRANT_MOUNT_BASE=/home/user/repos
```

Lancement de la machine virtuelle

Après avoir (éventuellement) réalisé les étapes ci-dessus, vous êtes prêt à lancer votre machine virtuelle. Pour cela, clonez le dépôt fun-boxes :

```
git clone https://github.com/openfun/fun-boxes
# le readme est plein d'instructions fort utiles
cat fun-boxes/README.rst
```

Lancez votre machine virtuelle :

```
cd fun-boxes/releases/
vagrant up
```

En cas de problème, pensez à consulter le README dans lequel votre problème est peut-être déjà décrit.

Lancement d'un serveur web

Si vous avez correctement lancé votre machine virtuelle, vous pouvez maintenant vous y connecter via ssh et lancer un serveur web local :

```
##### Commande exécutée sur votre machine hôte
vagrant ssh
```

```
##### Commandes exécutées dans la VM
```

```
# La plupart des applications sont exécutées par l'utilisateur edxapp
sudo su edxapp
```

```
# Cette commande réalise à la fois l'installation des dépendances, la
# collecte des données statiques et le lancement de l'application LMS
fun lms.dev run
```

Ouvrez maintenant votre navigateur (de votre machine hôte) à l'adresse <http://127.0.0.1:8000> : vous devriez voir apparaître la page d'accueil de FUN. Win !

```
# Pour sauter les phases de vérification de l'environnement, vous pouvez
# exécuter à la place de la commande précédente :
fun lms.dev run --fast
```

```
# De même, dans un autre terminal, vous pouvez lancer le Studio/CMS :
fun cms.dev run --fast
```

Le Studio/CMS est alors visible à l'adresse <http://127.0.0.1:8001>.

Vous pouvez également lancer les tests associés à FUN :


```
# Notez que les settings de test sont différents de ceux de dev
fun lms.test test ../fun-apps/
fun cms.test test ../fun-apps/
```

Sous le capot, ‘fun’ est un raccourci permettant d’exécuter une variété de commandes. Pour plus d’informations, consultez la documentation de fun-cmd : <https://github.com/openfun/fun-cmd>

Le forum

Le forum fonctionne avec un service REST Ruby qui utilise Mongo pour stocker les messages, Elasticsearch pour les indexer et un client Django qui se trouve dans le dépôt **edx-platform**.

Pour lancer le service forum dans un terminal :

```
sudo su forum
ruby app.rb -p 18080
```

Open edX et les différents types d’activités

Une activité Open edX peut aller de la simple page HTML aux quiz et évaluations par les pairs.

Pour qu’un MOOC ait du succès, il est recommandé de varier les types d’activités afin d’éviter l’aspect répétitif et permettre au cours d’être suivi avec plus d’engagement.

Lorsqu’un type d’activité n’est pas disponible directement dans Open edX, plusieurs options s’offrent aux développeurs pour les y ajouter (nous n’en décrivons que 2 ici) :

- Le **Xblock** : une extension en python qui doit être installée sur le serveur qui héberge la plateforme. C’est probablement la meilleure solution si vous avez accès au serveur.
- Le JS-Input **JS-Input** : probablement moins flexible en terme de possibilités offertes, mais probablement la réponse à de nombreux besoins. Le JS-Input a l’avantage d’être une extension dont l’installation se fait directement dans un cours sans nécessiter un accès au serveur.

[Options for Extending the edX Platform](#)

Mon premier Xblock ‘Hello Student !’

Les XBlocks sont les briques avec lesquelles sont construits les cours : il existe des XBlock pour afficher des vidéos dans le cours, pour y insérer des quiz, pour permettre des discussions de forum, ou même pour exécuter des lignes code. Un Xblock est donc une petite application web indépendante et réutilisable qui s’intègre dans edX, elle dispose de sa propre couche de persistance, et est constituée de vues Python à exécuter sur le serveur et de fichiers statiques HTML, CSS et Javascript pour le client. edX met à disposition un SDK qui aide à la création de XBlocks. Ainsi, il est possible commencer le développement de vos modules XBlock sans avoir à installer la plateforme edX.

Vous trouverez dans ce guide, les instructions pour installer le SDK et pour créer votre premier XBlock.

Installer le xblock sdk depuis le dépôt Github

Création de l’environnement virtuel :

```
mkdir venvs/  
virtualenv venvs/xblock-sdk  
source venvs/xblock-sdk/bin/activate
```

Installation du xblock sdk :

```
git clone https://github.com/edx/xblock-sdk.git  
cd xblock-sdk/  
make install  
python manage.py syncdb
```

Lancer le serveur de développement

```
python manage.py runserver 0.0.0.0:8010
```

Maintenant depuis votre navigateur allez à cette adresse 127.0.0.1 :8010. Si tout va bien la page suivante devrait apparaître :

Créons la structure de notre xblock

```
python script/startnew.py
```

XBlock scenarios

[XBlock Acid single block test](#)

[XBlock Acid Parent test](#)

[A little HTML](#)

[problem with thumbs and textbox](#)

[three problems 2](#)

[three thumbs at once](#)

Reset State

Le script demande d'abord un nom court pour notre xblock, choisissons 'hellostudent'. Ensuite rentrons le nom de classe 'HelloStudentXBlock'

Nous avons maintenant un dossier 'hellostudent' contenant la structure du XBlock.

Afficher 'Hello student'

Ouvrons le fichier `hellostudent/static/html/hellostudent.html` et remplaçons son contenu par :

```
<div class="hellostudent_block">
  <p>Hello Student !</p>
</div>
```

Enregistrer notre xblock dans le workbench.

Pour afficher notre xblock il est nécessaire de l'installer dans l'environnement de travail, le 'workbench'. L'installation est contrôlée par le fichier `setup.py` qu'il faudra modifier pour l'adapter à nos besoins.

```
# Se mettre dans l'environnement virtuel avant l'installation du paquet.
cd hellostudent/
pip install -e .
```

Ici, nous travaillons dans le contexte du SDK, mais sachez que ce même principe utilisant `pip install` est utilisé pour installer un XBlock dans la plateforme edX.

Vous devriez maintenant avoir un environnement minimal complet.

Pour rappel, la commande pour démarrer le serveur :

```
python manage.py runserver 0.0.0.0:8010
```

Voici ce que vous devriez voir :

Liens utiles

- La documentation officielle mais en cours de construction : <http://xblock.readthedocs.org/en/latest/>
- Un tutoriel pour lire et enregistrer des vidéos depuis un xblock : <http://opencraft.com/doc/edx/xblock/tutorial.html>
- Une liste des xblocks déjà existants : <https://github.com/edx/edx-platform/wiki/List-of-XBlocks>

JS-Input

Introduction

Le JS-Input est une spécificité d'OpenedX permettant d'étendre les types d'activités disponibles sur la plateforme.

Une activité JS-Input c'est : une page HTML avec un peu de Javascript !

Dans ce document nous allons expliquer comment construire une simple application JS-Input assez générique pour comprendre les mécanismes de base.

Les mécanismes de base

Tout d'abord, voici à quoi ressemble un problème de ce type dans studio :

Les paramètres de l'activité dans studio sont les suivants :

```
{% highlight xml %}
```

```
{% endhighlight %}
```

Les parties importantes de ce programme d'exemple, sont : - Le tag 'jsinput' qui définit la page à afficher (static/jsinput.html) - Le tag 'customresponse' qui



[XBlock Acid single block test](#)

[XBlock Acid Parent test](#)

[All Scopes](#)

[filethumbs](#)

[HelloStudentXBlock](#)

[Hello World](#)

[A little HTML](#)

[problem with thumbs and textbox](#)

[three problems 2](#)

[three thumbs at once](#)

Reset State

edX XBlock: HelloStudentXBlock

Hello Student !

Acid Aside for hellostudentxblock.hellostudent.d0.u0

Acid Aside for hellostudentxblock.vertical_demo.d0.u0

Database

Scenario State

Reset State

Block

<VerticalBlockWithMixins @42F9 name=None, parent=None, tags=[], children=[u'hellostudentxblock.hellostudent.d0.u0']>

Scenario

<vertical_demo>
<hellostudent/>
</vertical_demo>

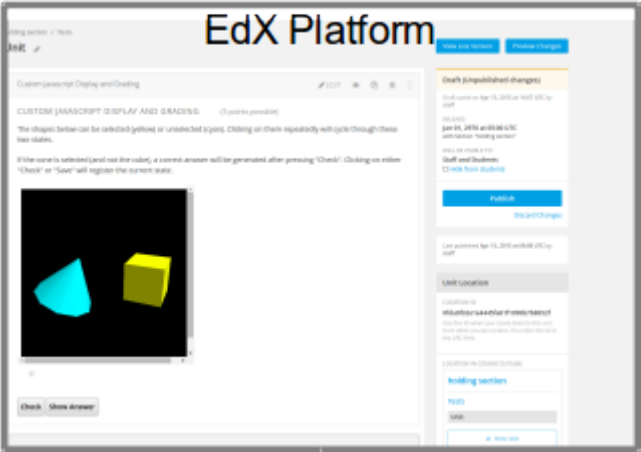


FIGURE 1 – Exemple d’activité dans studio

correspond au problème en lui-même. - Le script ‘loncapa/python’ qui permet d’analyser les réponses du problème et de retourner une note.

On peut en déduire les étapes clés dans l’instanciation d’une activité JS-Input :

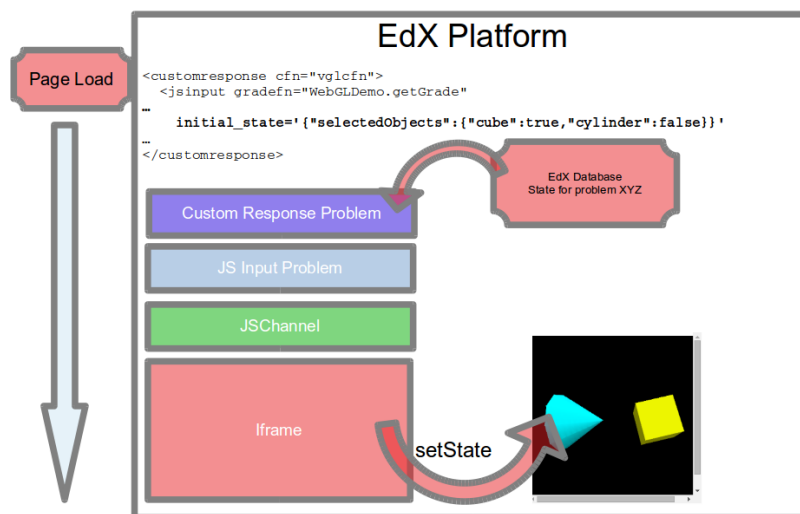
- Le chargement de l’activité et restauration de l’état initial : **set_statefn**
- Les actions de vérification du problème côté Open edX : **grade_fn**
- Les actions de changement : de note ou d’état **get_statefn** et **grade_fn**

Chargement de l’activité et initialisation de l’état

L’activité se charge dans la page de cours et utilise différents modules internes à Open edX.

Le module principal est “Custom Response Problem” qui est le module générique dans Open edX, permettant d’évaluer une réponse de manière programmatique. L’autre module est appelé JSChannel et permet à l’application JS-Input de communiquer avec Open edX. Nous allons revenir en détail vers ces deux modules dans un autre chapitre.

Pour l’instant occupons-nous du processus décrit sur ce schéma :



Lorsque la page se charge, Open edX retrouve le dernier état de l’application pour un utilisateur donné. Cet état se présente sous la forme d’une information codée en JSON. Le format de cette information est particulière à l’application JS-Input (seule elle la comprend en réalité). Sa signification est définie par le créateur de l’activité.

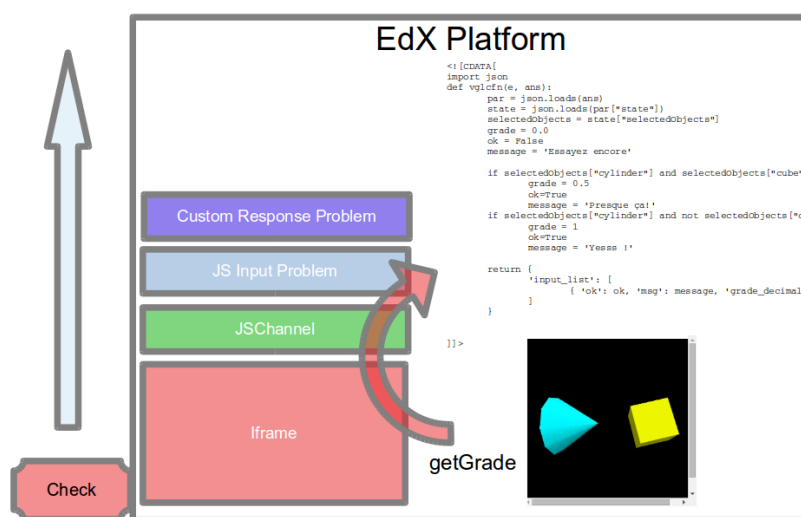
Si aucun “état” (JSON) pour l’utilisateur n’est trouvé et que l’on a spécifié un état initial, celui-ci est chargé et présenté à l’application JS-Input par un appel à la fonction “setState”.

Vérification du problème coté Open edX

La routine de vérification d'un problème est activée par l'appui de l'utilisateur sur le bouton "Vérifier" (ou "Check" en Anglais). C'est seulement cette action qui déclenchera la séquence de vérification.

Ce qui se passe :

- Le conteneur JS Input Problem envoie un "Get grade" pour récupérer une l'information d'état de l'activité encodée en JSON. Il appellera aussi la fonction "Get State" si elle existe pour stocker l'état actuel de l'utilisateur.
- L'information passe à travers toutes les couches logicielles (JSChannel, JS Input Problem) et vers edX
- Le script python intégré à l'activité JS-Input dans Open edX est lancé pour vérifier le résultat, et renvoie une information sous forme de note
- Le résultat est renvoyé vers le serveur edX



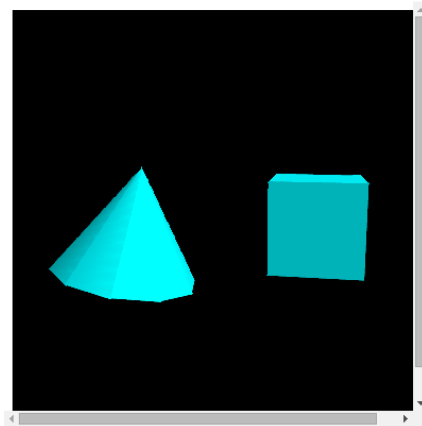
Ensuite le résultat est stocké dans la base de donnée Open edX avec :

- des informations sur le temps exact de soumission,
- un objet **correct_map** qui permet de stocker le status (correct ou non) de la réponse après analyse par le script python de l'exercice.

Vous pouvez voir l'historique des soumissions grâce au bouton "Historique des soumissions" situé au dessous de l'activité (seulement accessible par l'enseignant).

Cet historique va donner des résultats comme ceux-ci (application d'exemple Javascript) :

#4: 2015-05-11 20:46:34+00:00 (Europe/Paris time)



Vérifier

Afficher la réponse

HISTORIQUE DES SOUMISSIONS

INFO DE DÉBOGAGE POUR L'ÉQUIPE PÉDAGOGIQUE

Score: 1.0 / 1.0

```
{
  "attempts": 1,
  "correct_map": {
    "i4x-FUN-FUN101-problem-2d1cf6dd9012475ebf3d6295ccb1da72_2_1": {
      "correctness": "correct",
      "hint": "",
      "hintmode": null,
      "msg": "",
      "npoints": 1,
      "queuestate": null
    }
  },
  "done": true,
  "input_state": {
    "i4x-FUN-FUN101-problem-2d1cf6dd9012475ebf3d6295ccb1da72_2_1": {}
  },
  "last_submission_time": "2015-05-11T20:46:34Z",
  "seed": 1,
  "student_answers": {
    "i4x-FUN-FUN101-problem-2d1cf6dd9012475ebf3d6295ccb1da72_2_1": "{\"answer\":\"{\\\\\\\\\"cylir
  }
}
```

Mécanismes de retour d'information

Il existe un troisième mécanisme de retour d'information appelé `get_statefn`. Dans la pratique, on peut se baser sur le retour de la note (qui peut donner bien plus qu'un état de note, mais aussi une idée de l'état de l'application). Dans ce cas, on va pouvoir définir une fonction de l'application qui est appelée lorsque l'on requiert un statut sur l'application. Dans ce cas la réponse de l'application devra comporter deux champs : `answer` et `state`.

Exemple :

```
{
  "answer": "{\"cylinder\":true,\"cube\":false}\",
  "state": "{\"selectedObjects\":{\"cylinder\":true,\"cube\":false}}\"
}
```

Les modules

Custom Response Problem et JS Input Problem

Ces deux types de problèmes sont des modules permettant de vérifier la réponse utilisateur par un petit script python avant l'enregistrement réel sur Open edX. Ceci permet de faire pas mal de choses notamment de noter de manière plus souple tout en restant automatique.

La documentation est disponible ici : <https://github.com/Stanford-Online/js-input-samples>

JSChannel

JSChannel est un wrapper créé par Mozilla pour faciliter la communication entre pages et iframes (voir `window.postMessage` : <https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage>). La bibliothèque JS Channel facilite ce travail : <https://github.com/mozilla/jschannel>.

Trucs et astuces

Intégrer du JS Input directement de github

Lorsque l'on développe une extension, il est assez pratique d'avoir une version de l'application externe en cours sur un site externe. Sinon on est obligé de recharger les fichiers correspondants à chaque mise à jour.

Pour cela il est pratique d'utiliser le lien provenant directement de github sur les ressources : <https://rawgit.com/>

Faire une activité qui retourne une note différente de 0 ou 1

```
{% highlight xml %}
```

```
{% endhighlight %}
```

Liens utiles

- Documentation de l'activité JS Input : http://edx-partner-course-staff.readthedocs.org/en/latest/exercises_tools/custom_javascript.html
- Documentation identique mais orientée développeur : http://edxpdrllab.readthedocs.org/en/latest/course_data_formats/jsinput.html
- Custom Python evaluated problem (Version générique du JS Input) : http://edx-partner-course-staff.readthedocs.org/en/latest/exercises_tools/custom_python.html
- Stanford JS Input Samples : <https://github.com/Stanford-Online/js-input-samples>

Analytics

Introduction

FUN met à la disposition des participants au hackathon une quantité de logs extraits de ses machines de productions à fins d'analyse.

Téléchargement des fichiers de logs

TODO

Source des logs

Les logs proviennent des appels à `tracker.emit` qui parsèment le code d'edX et de FUN : <https://github.com/edx/event-tracking/blob/0.2.0/eventtracking/tracker.py#L65>

Chaque évènement loggé se présente sous la forme d'un blob JSON contenant au moins un champ `time`.

Format des logs

Les logs fournis par FUN sont anonymisés, ce qui signifie que les champs `email`, `address`, etc. ont été retirés des blobs JSON. Par ailleurs, le champs `username` a été chiffré à l'aide d'une méthode de chiffage à sens unique :

```
encrypted_username = hmac.new(secret_key, username, hashlib.sha256).hexdigest()
```

Analyse des logs à l'aide de Elasticsearch

Les logs fournis par FUN se prêtent particulièrement bien à l'analyse via Elasticsearch. Si vous décidez de charger les logs fournis dans un cluster Elasticsearch, nous vous recommandons d'installer la pile ELK : Elasticsearch + Logstash + Kibana.

- Elasticsearch est le moteur d'indexation et de recherche de vos données.
- Kibana est le frontend qui vous permettra de visualiser vos données dans le navigateur.
- Logstash permet d'envoyer vos logs à Elasticsearch en les convertissant en événements au format ad-hoc.

Installation

L'installation des trois composants de la stack ELK est bien documentée :

- <https://www.elastic.co/downloads/elasticsearch>
- <https://www.elastic.co/downloads/logstash>
- <https://www.elastic.co/downloads/kibana>

Envoi des logs vers Elasticsearch

Une fois que vous avez correctement installé Logstash et Elasticsearch, vous pouvez insérer les logs de FUN dans Elasticsearch à l'aide du fichier de configuration `logstash.conf` fourni dans ce dépôt :

```
cat fun_tracking_logs.log | logstash --config static/logstash.conf
```

Visualisation des résultats dans Kibana

Après avoir inséré quelques événements dans Elasticsearch, vous pouvez lancer Kibana et observer ces événements en ouvrant <http://localhost:5601> dans votre navigateur. N'oubliez pas de sélectionner un intervalle de temps couvert par les logs (en haut à droite).

Réaliser des requêtes manuelles sur Elasticsearch

Vous pouvez souhaiter réaliser des requêtes complexes sur Elasticsearch et en récupérer le résultat brut au format JSON sans passer par Kibana. Pour ça, le mieux est de :

1. créer une requête via Kibana, dans l'onglet "Discover".
2. récupérer cette requête au format JSON, en repliant le graphe de résultats, puis sous l'onglet "Request". Par exemple :

```
{ "size" : 500, "sort" : { "@timestamp" : "desc" }, "query" : { "filtered" : { "query" : { "query_string" : { "query" : "*", "analyze_wildcard" : true } } } }
```
3. Copier-coller cette requête dans un fichier `query.json`, puis réaliser la requête à l'aide du script fourni dans ce dépôt :

```
static/es.py query.json > result.json
```

A quoi ça ressemble ?

A la fin de ce document vous disposez de quelques photos d'écran annotées de quelques phrases pour décrire la manière dont Open edX se présente de manière standard.

Ce que nous pouvons retenir : * Le LMS est beaucoup plus configurable que le CMS/Studio et les thèmes ne marchent **que pour le LMS** grâce au Thème Stanford (voir prochain chapitre).

- Il reste quelques soucis d'ergonomies et de présentation à l'intérieur du cours. Cela est beaucoup plus difficile à adresser de manière efficace et compatible avec les mise à jour. Néanmoins nous n'allons pas poser de limites aux changements tant que cela se passe à l'extérieur du code "source" d'Open edX (à la manière Stanford Thème). Par exemple :

1. Le contenu d'un cours : limité à une taille maximale en largeur et ne met pas assez en valeur le contenu.
2. Les forums ont gagné en ergonomie mais restent toujours peu intuitifs (notamment pour l'enseignant, il est parfois difficile de s'y repérer)

Modifier l'apparence d'Open edX

Notez bien que ces solutions sont valides seulement pour le LMS.

Solution 1 : Thème Stanford

Solution à l'origine proposée par Stanford et utilisée par FUN et IonisX.

Le document original est ici : <https://github.com/edx/edx-platform/wiki/Stanford-Theming> et là <https://github.com/Stanford-Online/edx-theme>.

Les étapes sont : 1. Modifiez le fichier de configuration `/edx/app/edxapp/lms.env.json` et les variables `FEATURES.USE_CUSTOM_THEME`, `THEME_NAME` et `PLATFORM_NAME` 2. Mettez votre thème dans `/edx/app/edxapp/themes/`

Pas à pas avec le “Stanford Theme”

1. Clonez le projet <https://github.com/Stanford-Online/edx-theme> dans le répertoire `/edx/app/edxapp/themes/`
2. Renommez le fichier `__default.scss` vers le nom de votre thème (ici `edx-theme`) :

```
mv /edx/app/edxapp/themes/edx-theme /edx/app/edxapp/themes/default
```

3. Changez votre configuration

Exemple de configuration `lms.env.json` (extrait) :

```
{% highlight %} “FEATURES” : { “AUTH_USE_OPENID_PROVIDER” :  
true, ... “USE_CUSTOM_THEME” : true }, ... “THEME_NAME” : “default”,  
{% endhighlight %}
```

Redémarrez votre serveur pour recompiler les “assets” (parties statiques, fichiers css et javascripts).

Thème IonisX

Même procédure que le thème Stanford excepté qu’il faut cloner le thème IonisX [https://github.com/IONISx/edx-theme/](https://github.com/IONISx/edx-theme) La documentation est dans le `README.md`

Thème FUN

Le thème FUN est plus facilement installé grâce aux machines virtuelles disponibles (voir documentation d’installation).

<https://github.com/openfun/edx-theme> La documentation est dans le `README.md`

Liens Utiles

Thème Stanford (utilisé par FUN-MOOC)

<https://github.com/Stanford-Online/edx-theme> <https://github.com/edx/edx-platform/wiki/Stanford-Theming>

Exemples de thèmes

<https://github.com/edx/edx-platform/wiki/Sites-powered-by-Open-edX>

Documentation générale

<https://github.com/edx/edx-platform/wiki/Javascript-standards-for-the-edx-platform>

<https://github.com/edx/edx-platform/wiki/Alternate-site-for-marketing-links>

Photos d'écran et suggestions

Ecran d'accueil et écrans externes au cours

Facile à mettre à jour dans le thème par le (header et footer). On peut vraiment améliorer l'ergonomie ici. Notez bien qu'il existe aussi des pages statiques que l'on peut changer aussi facilement (voir <https://github.com/edx/edx-platform/wiki/Alternate-site-for-marketing-links>)

L'écran d'accueil

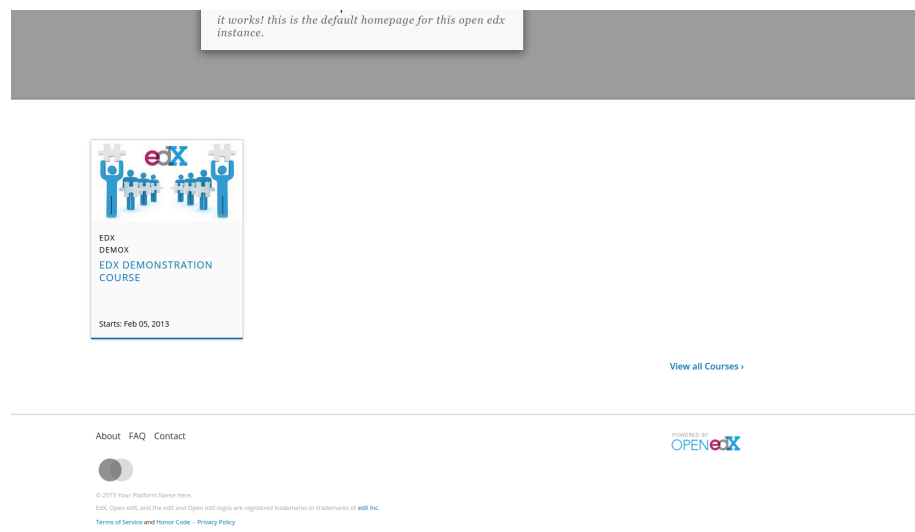


FIGURE 2 – L'écran d'accueil

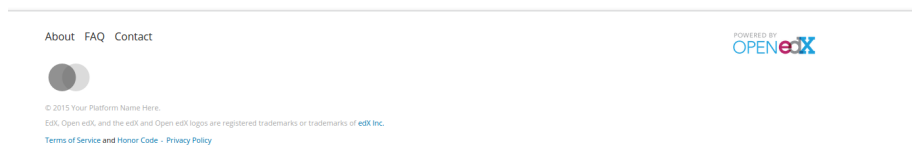


FIGURE 3 – Le footer (à changer)

Le footer

Le tableau de bord étudiant et sa liste des cours

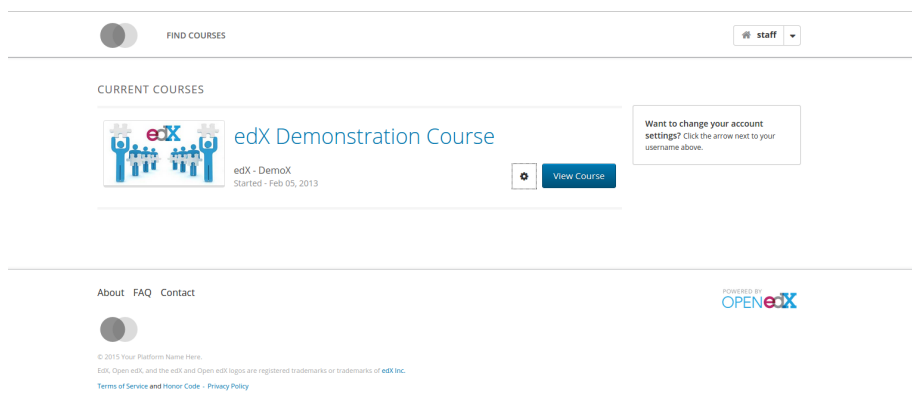


FIGURE 4 – Le tableau de bord étudiant (liste des cours)

Pages d'inscription et de connexion

Là aussi un travail d'ergonomie peut être fait :


S'inscrire


Se connecter

Le syllabus d'un cours

Regardez les différents exemples de thèmes (IonisX et FUN) pour voir à quoi cela ressemble sur un site un peu plus fournis en contenu.

Notez qu'edX (edx.org) n'utilise pas edX pour afficher ses syllabus mais qu'il y a de très bonnes idées dans leur présentation :


[REGISTER NOW](#)
[Sign in](#)



WELCOME!

register below to create your your platform name here account

Please complete the following fields to register for an account.
Required fields are noted by **bold text and an asterisk (*)**.

E-mail *

Full Name *

Needed for any certificates you may earn

Public Username *

Will be shown in any discussions or forums you participate in (cannot be changed later)

Password *

Highest Level of Education Completed

Gender

Year of Birth

Mailing Address

Please share with us your reasons for registering with Your Platform Name Here

ALREADY REGISTERED? [Log in](#)

WELCOME TO YOUR PLATFORM NAME HERE

Registering with Your Platform Name Here gives you access to all of our current and future free courses. Not ready to take a course just yet? Registering puts you on our mailing list - we will update you as courses are added.


NEXT STEPS

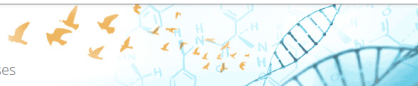
As part of joining Your Platform Name Here, you will receive an activation email. You must click on the activation link to complete the process. Don't see the email? Check your spam folder and mark Your Platform Name Here emails as 'not spam'. At Your Platform Name Here, we communicate mostly through email.

NEED HELP?

Need help in registering with Your Platform Name Here? View our FAQs for answers to commonly asked questions. Once registered, most questions can be answered in the course specific discussion forums or through the FAQs.

FIGURE 5 – S’inscrire


[REGISTER NOW](#)



PLEASE LOG IN

to access your account and courses

E-mail *

This is the e-mail address you used to register with Your Platform Name Here

Password *

[Forgot password?](#)

☐ **Remember me ***

[Log into My Your Platform Name Here Account](#) + [Access My Courses](#)


NOT ENROLLED?

[Sign up for Your Platform Name Here today!](#)

NEED HELP?

Looking for help in logging in or with your Your Platform Name Here account? View our help section for answers to commonly asked questions.

About [FAQ](#) [Contact](#)

POWERED BY 

© 2015 Your Platform Name Here.
edX, Open edX, and the edX and Open edX logos are registered trademarks or trademarks of edX Inc.
[Terms of Service and Honor Code](#) - [Privacy Policy](#)

FIGURE 6 – Se connecter

Le syllabus d'un cours

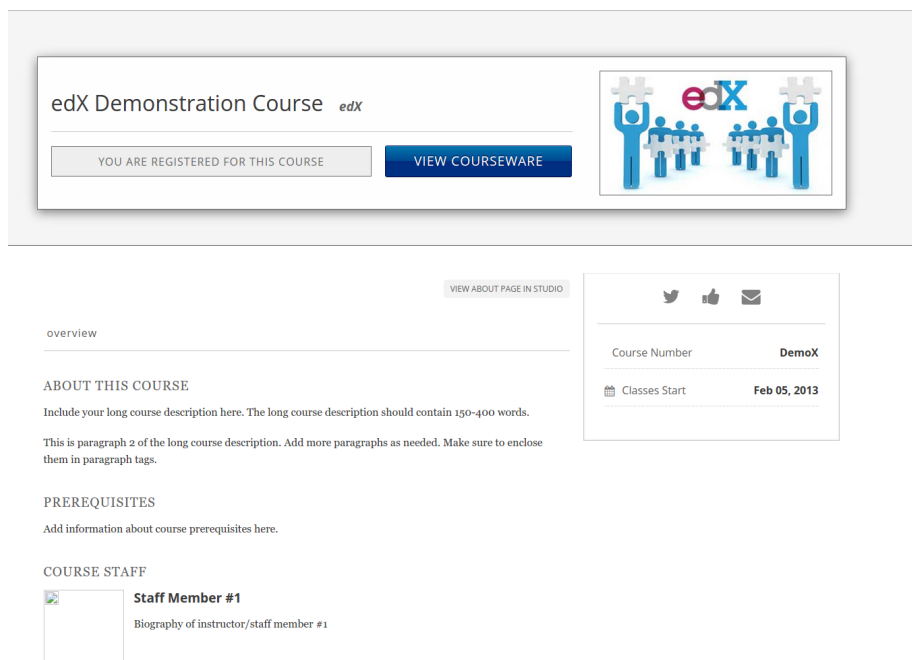


FIGURE 7 – Le syllabus d'un cours

Intérieur d'un cours

Là aussi des améliorations peuvent être pensées. Sachez tout de même que cela reste plus difficile à maintenir sur une plateforme en production à cause des nombreuses mises à jour.

Le forum

Le contenu du cours

Administration

MySQL

Vous pouvez accéder au shell MySQL avec l'utilisateur `edxapp001`, le mot de passe est `password` :

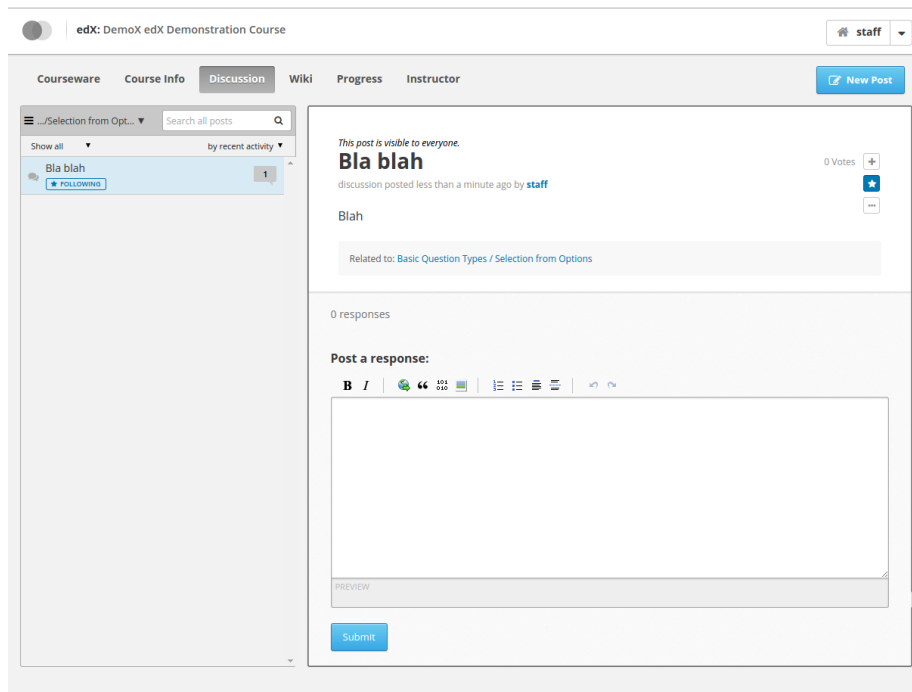


FIGURE 8 – Le forum

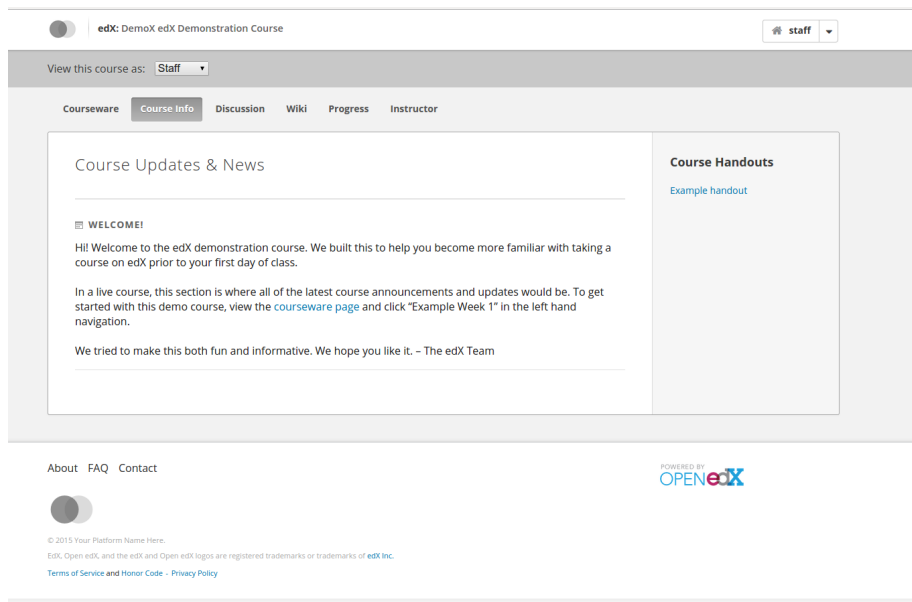


FIGURE 9 – Le cours

```
mysql -u edxapp001 -p
```

Vous pouvez aussi y accéder via Django et la commande `fun` :

```
fun lms.dev dbshell
```

MongoDB

Quelques commandes pour accéder aux collections Mongo :

```
mongo
MongoDB shell version: 2.6.6
connecting to: test
> show dbs
admin                0.078GB
cs_comments_service_development 0.078GB
edxapp               0.453GB
local                0.078GB
> use edxapp
switched to db edxapp
> show collections
assetstore
fs.chunks
fs.files
modulestore
system.indexes
> db.modulestore.find()
```

Les logs

Les fichiers de logs propres à edX se trouvent dans `/edx/var/log/`

Les logs applicatifs studio et lms :

```
sudo tail -f /edx/var/log/lms/edx.log
sudo tail -f /edx/var/log/cms/edx.log
```

Les tracking logs qui regroupent le comportement des utilisateurs du lms :

```
/edx/var/log/tracking
```