Présentation générale

Table des matières

Introduction

Cette documentation est la version pdf du site disponible sur http://openfun.github.io/hackathon/ et dans le dépôt github correspondant.

Présentation générale

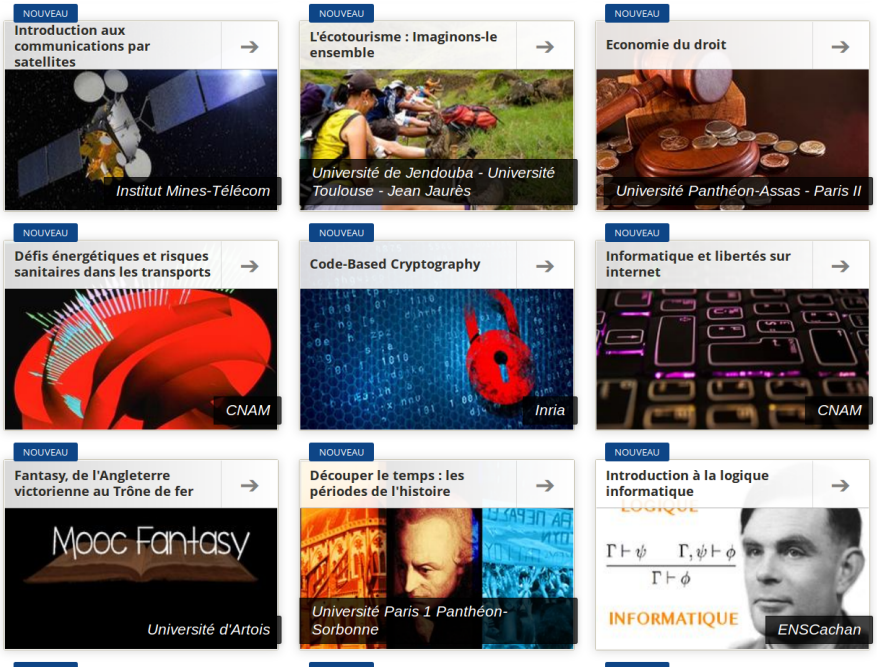
edX est une platforme web qui permet de délivrer des cours en ligne ouverts à tous, les MOOCs - Massive Online Open Courses.

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 barêmes 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'inscrirent, 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 dévelopement.



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) :

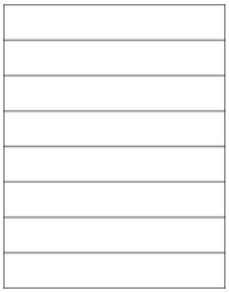
Et les professeurs concoivent ces moocs depuis un *CMS* (Content Management System), appelé aussi *Studio* :

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.

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

✖

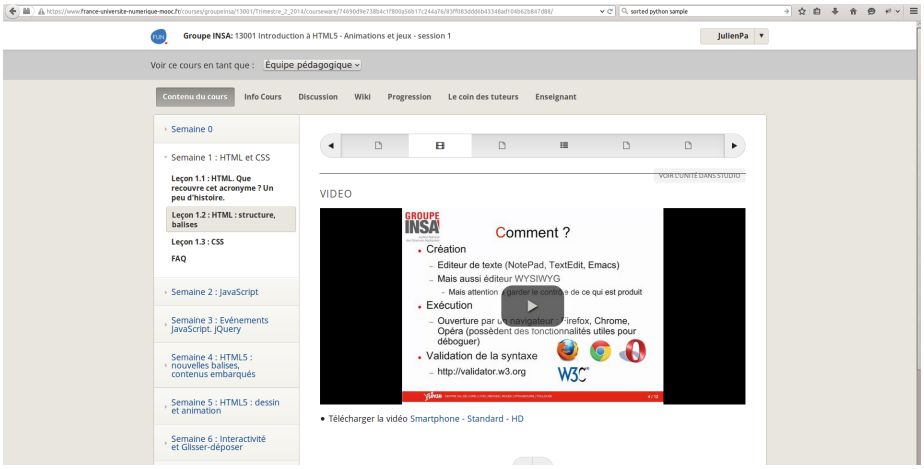
br

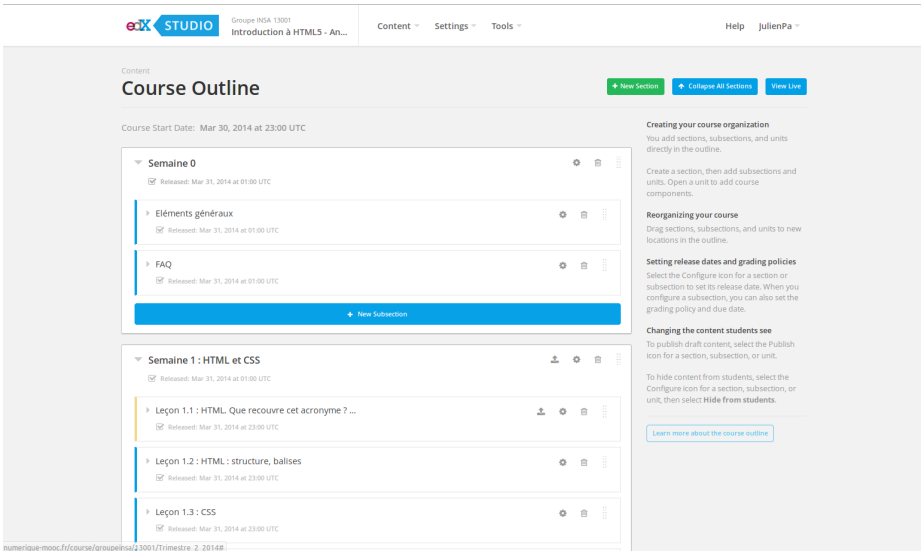
✔

em

gras ✖

strong


Groupe INSA: 13001 Introduction à HTML5 - Animations et jeux - session 1
JulienPa
Voir ce cours en tant que : Équipe pédagogique
Contenu du cours
Info Cours
Discussion
Wiki
Progression
Le coin des tuteurs
Enseignant
Semaine 0
Semaine 1 : HTML et CSS
Leçon 1.1 : HTML. Que recouvre cet acronyme ? Un peu d'histoire.
Leçon 1.2 : HTML : structure, balises
Leçon 1.3 : CSS
FAQ
Semaine 2 : JavaScript
Semaine 3 : Evénements JavaScript, jQuery
Semaine 4 : HTML5 : nouvelles balises, contenus embarqués
Semaine 5 : HTML5 : dessin et animation
Semaine 6 : Interactivité et Glisser-déposer
VIDEO
Comment ?
Création
Editeur de texte (NotePad, TextEdit, Emacs)
Mais aussi éditeur WYSIWYG
Exécution
Validation de la syntaxe
http://validator.w3.org
Télécharger la vidéo Smartphone - Standard - HD


STUDIO
Groupe INSA 13001
Introduction à HTML5 - An...
Content
Settings
Tools
Help
JulienPa
Course Outline
New Section
Collapse All Sections
View Live
Course Start Date: Mar 30, 2014 at 23:00 UTC
Semaine 0
Released: Mar 31, 2014 at 01:00 UTC
Eléments généraux
FAQ
New Subsection
Semaine 1 : HTML et CSS
Leçon 1.1 : HTML. Que recouvre cet acronyme ? ...
Leçon 1.2 : HTML : structure, balises
Leçon 1.3 : CSS
Creating your course organization
Reorganizing your course
Setting release dates and grading policies
Changing the content students see
Learn more about the course outline

Prérequis

— Configuration recommandée : Ubuntu/Linux 14.04
— VirtualBox >= 4.3.12
— Vagrant >= 1.6.5

Version Open FUN

FUN - 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
```

Version edX (Birch)

Les étapes sont données en détail ici https://github.com/edx/configuration/wiki/edX-Developer-Stack

La préparation

Créer un répertoire et cloner les repositories principaux :

```
mkdir devstack
cd devstack
curl -L https://raw.githubusercontent.com/edx/configuration/master/vagrant/release/devstack/
```

```
git clone https://github.com/edx/edx-platform.git
git clone https://github.com/edx/cs_comments_service.git
vagrant plugin install vagrant-vbguest
vagrant up
```

Lancer la VM

Une fois que l'installation s'est bien déroulée, vous pouvez démarrer votre devstack en tapant :

```
vagrant ssh
sudo su edxapp
paver devstack lms &
paver devstack studio &
```

Comme pour l'installation de FUN, le forum est démarré par un :

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

Notes

Soucis avec la VM

Il a été constaté que parfois l'application était très lente voire se bloquait complètement. Ceci est dû à un problème de DNS dans virtual box. Voir http://stackoverflow.com/questions/28562968/django-1-4-18-dev-server-slow-to-respond-under-virtualbox/30356662#30356662. La solution est de rajouter 10.0.2.2 10.0.2.2 dans votre /etc/hosts

Pour d'autres trucs et astuces voir la page : https://github.com/openfun/fun-boxes

La commande FUN

La commande 'fun' vous donne accès directement à la commande de base appelée 'manage.py'. Tapez par exemple :

```
fun lms.dev --help
```

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.

Bien que l'offre soit assez vaste, "beaucoup" n'est souvent pas assez pour l'ensemble des acteurs de la plateforme.

Pour répondre à la demande des MOOC, il faut souvent créer de multiple types d'activités afin d'éviter l'aspect répétitif et permettre au cours d'être suivi avec plus d'engouement. Il ne faut pas être limité à un choix toujours trop réduit de types de quiz.

Open edX propose de résoudre ce problème en mettant à portée du développeur/concepteur deux technologies :

— 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 nécéssiter un accès au serveur.

Mon premier Xblock 'Hello Student !'

Les XBlocks enrichissent les contenu de 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. 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 platforme Edx.

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

Installer le xblock sdk depuis le dépot Github

Création de l'environnement virtuel :

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

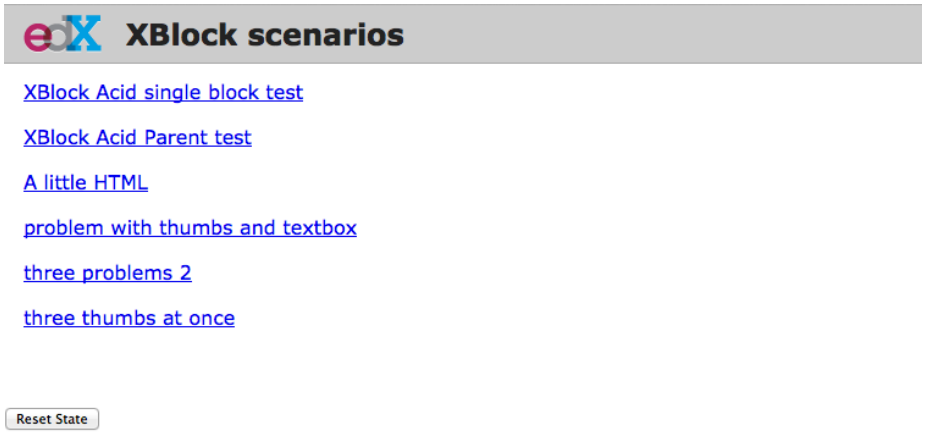
Installation du xblock sdk :

```
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
```

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 :

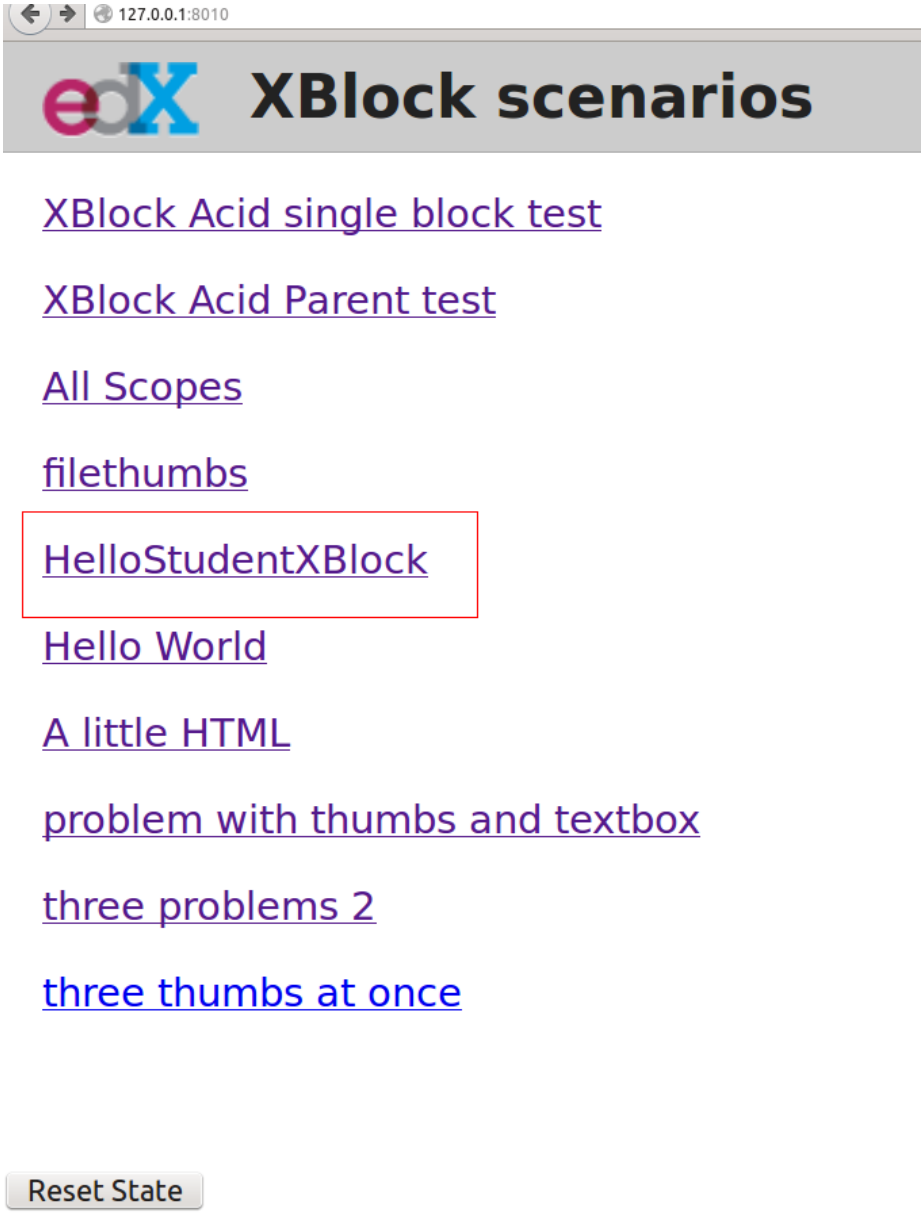
Liens utiles

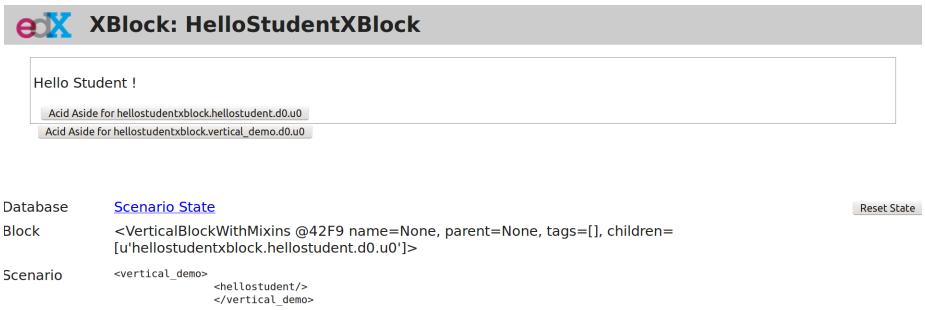
- La documentation officielle mais en cours de construction : http://xblock.readthedocs.org/en/latest/
- Un tutoriel pour lire et enregistrer des vidéos depuis un xblock : http://opencraft.com/doc/edx/xblock/tutorial.html
- Une liste des xblocks déjà existants : https://github.com/edx/edx-platform/wiki/List-of-XBlocks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

JS-Input

Introduction

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


127.0.0.1:8010
XBlock scenarios
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



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 :

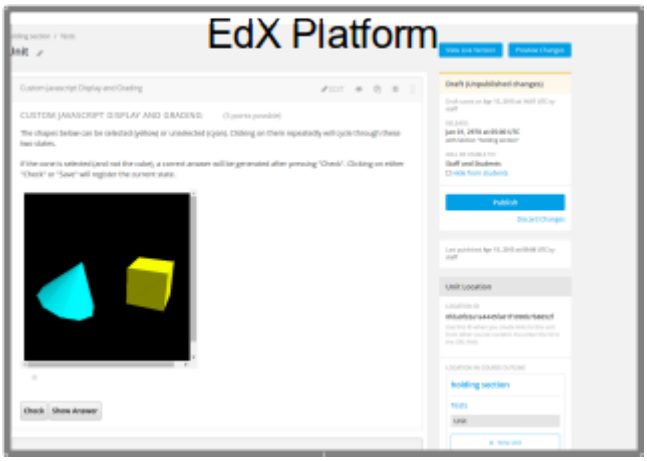


FIGURE 1 – Exemple d'activité dans studio

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 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 :

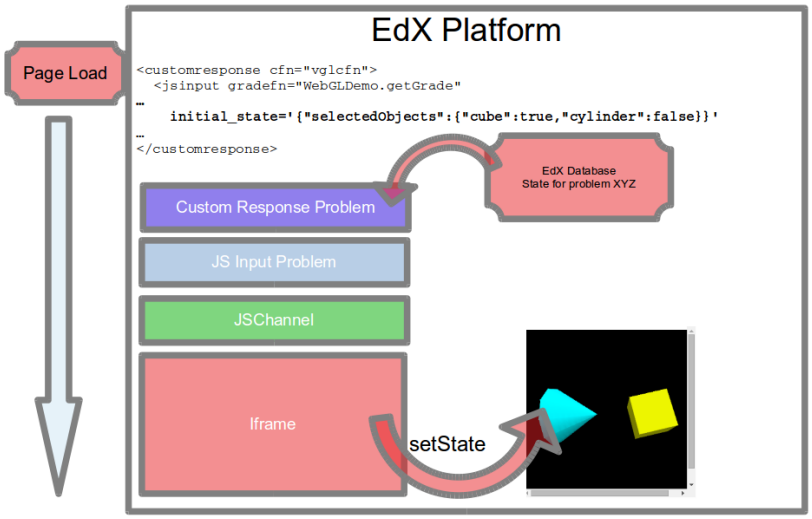
— Le chargement de l'activité et restauration de l'état intial : `set_statefn`
— Les actions de vérification du problème côté Open edX : `gradefn`
— Les actions de changement : de note ou d'état `get_statefn` et `gradefn`

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.

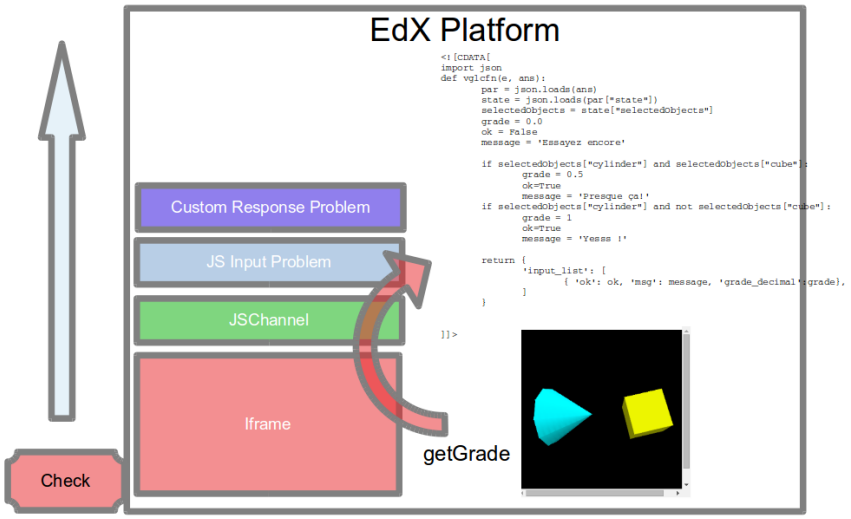
Si aucun "état" (JSON) pour l'utilisateur n'est trouvé et que l'on a spécifié un état initial, celui-ci est chargé et présenté à l'application JS-Input par un appel à la fonction "setState".

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



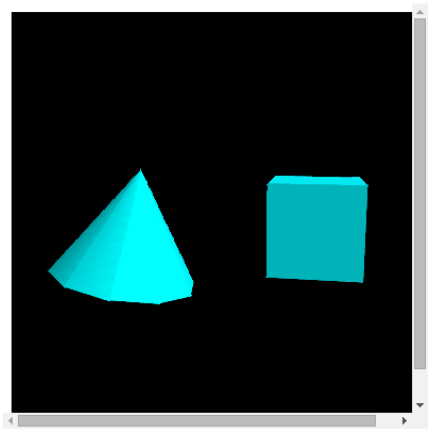
Ensuite le résultat est stocké dans la base de donnée Open edX avec :

- 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 diponible 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 gihub sur les resources : 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://edxpdrlab.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 chiffrage 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 évènements 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 évènements dans ElasticSearch, vous pouvez lancer Kibana et observer ces évènements 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 Standford (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 Standford 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 Standford

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 "Standford Theme"

1. Clonez le projet https://github.com/Stanford-Online/edx-theme dans le répertoire /edx/app/edxapp/themes/
2. Renommez le fichier _default.sccs 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 json %} "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/ 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'accueuil 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

FIGURE 2 – L'écran d'accueil

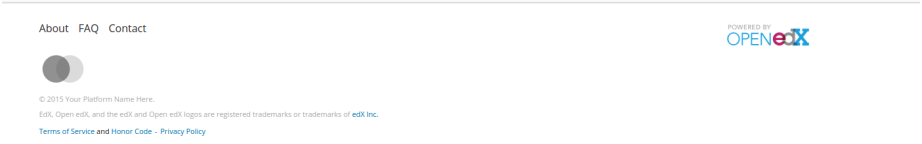


FIGURE 3 – Le footer (à changer)

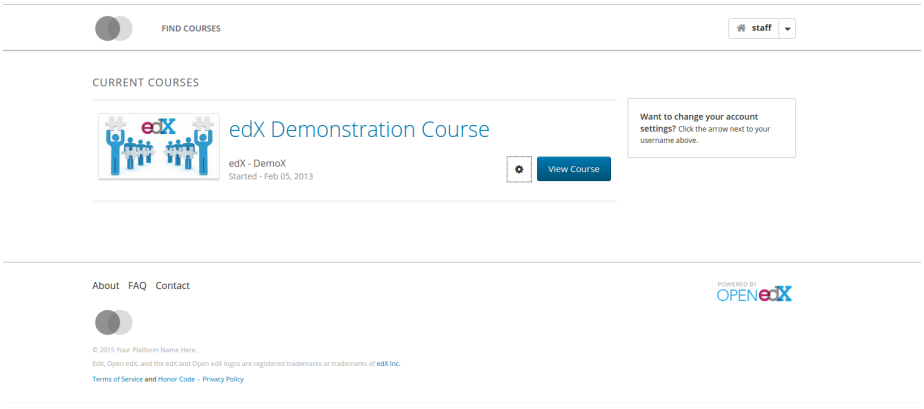


FIGURE 4 – Le tableau de bord étudiant (liste des cours)

L'écran d'accueil

Le footer

Le tableau de bord étudiant et sa liste des cours

Pages d'inscription et de connexion

Là aussi un travail d'ergonomie peut être fait :

S'inscrire

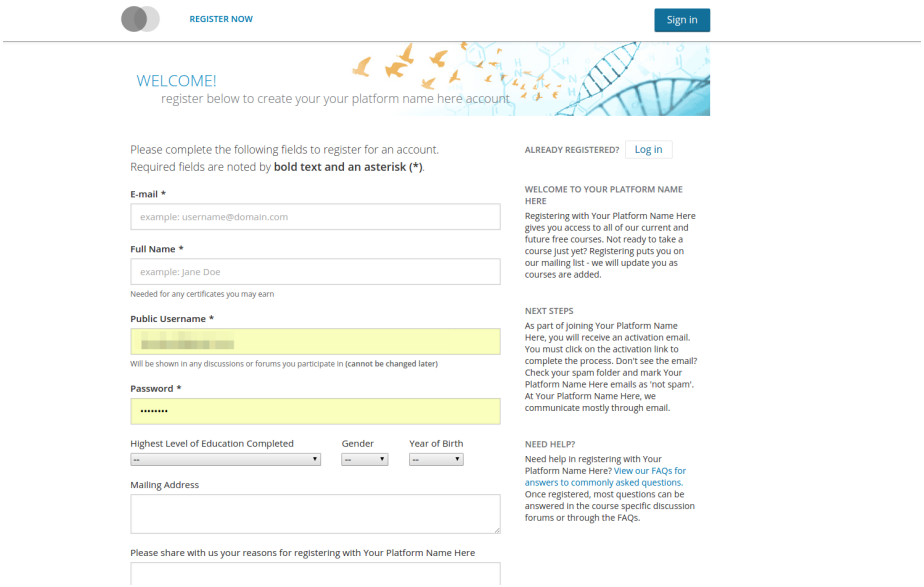


FIGURE 5 – S'inscrire

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 affichers ses syllabus mais qu'il y a de très bonnes idées dans leur présentation :

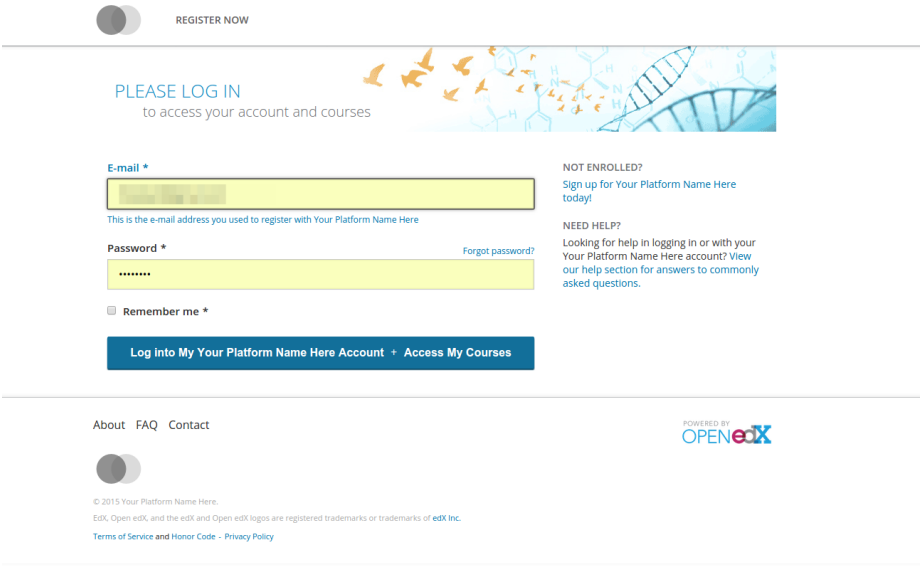


FIGURE 6 – Se connecter

Le syllabus d'un cours

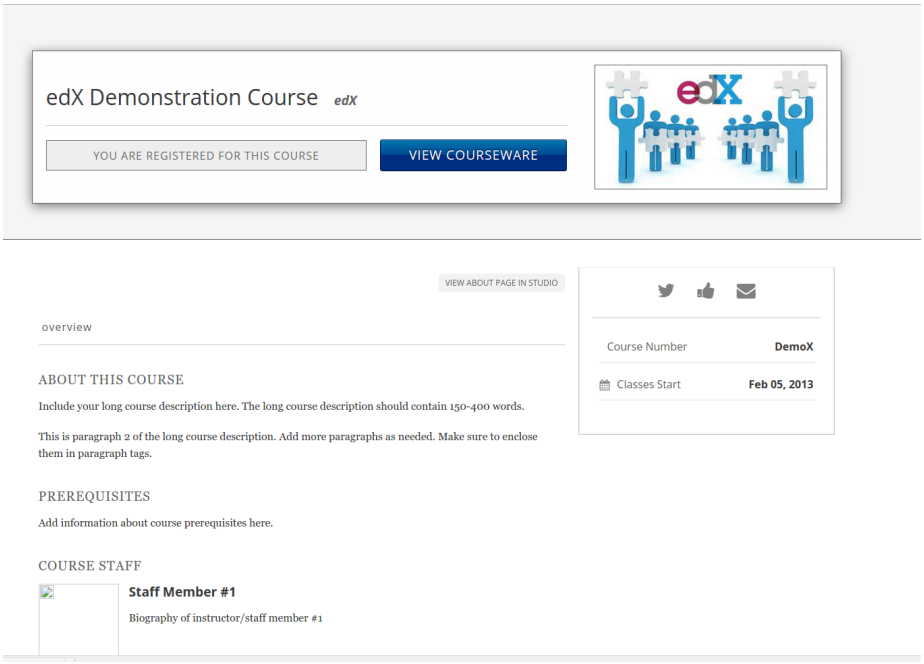


FIGURE 7 – Le syllabus d'un cours

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

Les MOOC et usages mobiles

Selon plusieurs sources statistiques (notamment Gartner), l'utilisation des site webs par des outils "mobiles" (tablettes, téléphonne portable) est en train de dépasser celle par les PC traditionnel.

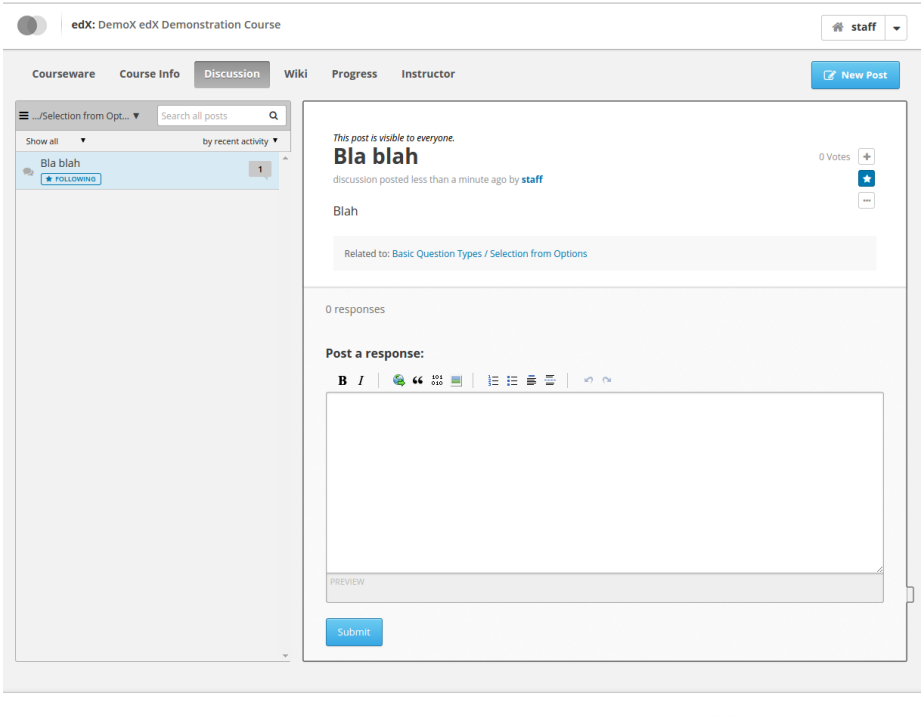


FIGURE 8 – Le forum

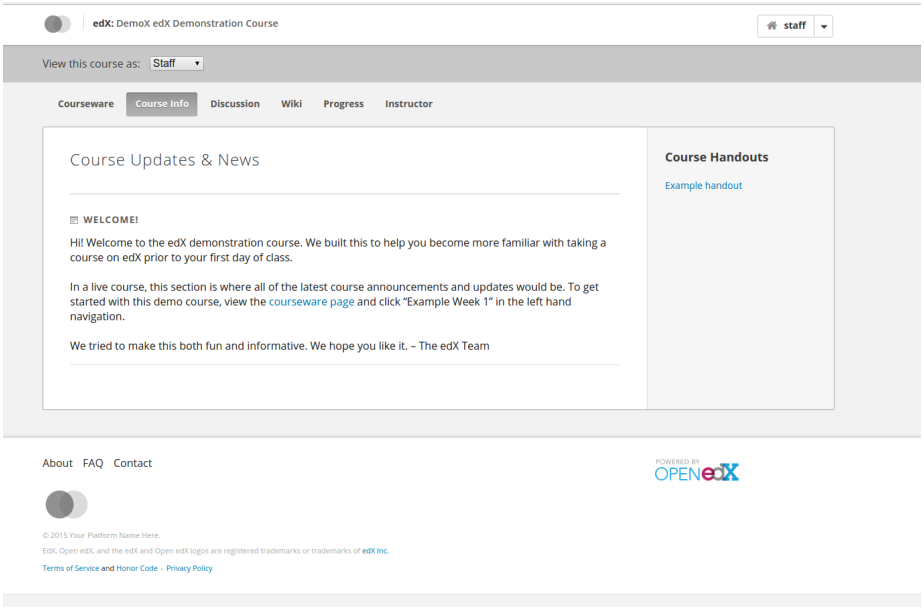


FIGURE 9 – Le cours

Il est alors évident que créer une application mobile pour suivre son MOOC est devenue nécessaire. Open edX, comme la plupart des plateformes de MOOC propose une version Mobile de ses applications.

Dans ce document nous allons voir comment installer l'application mobile sous Android et ainsi commencer à pouvoir développer sur application mobile.

Les prérequis

Une IDE pour Android et le SDK

Nous n'allons pas détailler ni conseiller ici d'IDE pour développer sur Android. La documentation essentielle est ici https://developer.android.com/sdk/index.html. Vous avez principalement le choix entre Android Studio et Eclipe.

Les deux ont leur avantages et inconvénients.

La seule chose à retenir ici est que les émulateurs du SDK ne doivent utilise ABI et non pas 'x86', sinon vous ne pourrez pas avoir un emulateur Android qui coexiste avec la VM de Open edX (Plus d'info ici).

L'application Mobile

L'application Mobile OpenedX est disponible ici : https://github.com/edx/edx-app-android

Pour l'installer, il suffit de suivre les instructions ici : http://edx-installing-configuring-and-running.readthedocs.org/en/latest/mobile.html

Les étapes sont : 1. Modification de lms.json pour activer les API mobiles 2. Création d'une clé OAuth

Quelques notes

Les vidéos

Pour l'instant l'application mobile n'est compatible qu'avec le lecteur vidéo natif d'edX.

Les API

Les API mobiles edX permettent d'accéder à l'intérieur des cours pour y rechercher les vidéos, les transcripts par exemple Mobile API Endpoints : - Détails pour un utilisateur donné - Page de Syllabus de cours - Les information / annonces du cours - Les vidéos...

Le lecteur vidéo : les bases

Le lecteur vidéo est très utilisé dans les MOOC. On compte en moyenne 30 ou 40 vidéos par MOOC.

Il existe plusieurs lecteurs vidéos disponibles pour Open edX. Le “natif” est celui fourni par la plateforme qui permet de lire des vidéos de Youtube (Lecteur Natif dont le manuel est ici).

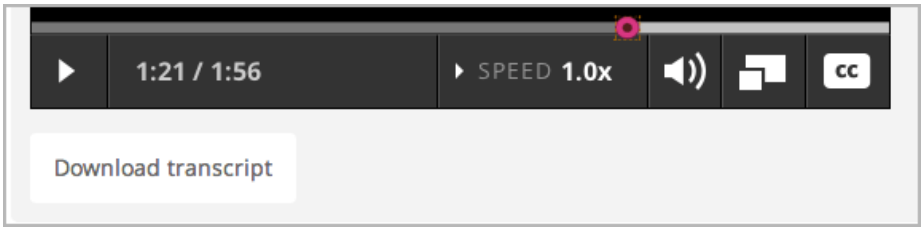
Les autres lecteurs vidéos, notamment celui utilisé par FUN-MOOC, ont des fonctionnalités équivalentes mais en utilisant un hébergement autre que Youtube.

Caractéristiques d’un lecteur vidéo

La plupart des lecteurs vidéos disponibles peuvent faire les choses suivantes :

1. Intégrer dans Open edX Studio des vidéos par leurs identifiants youtube ou directement par l’URL du fichier vidéo
2. Gérer des sous-titres en plusieurs langues (soit directement par le lecteur soit par l’hébergeur vidéo, cas de FUN-MOOC).
3. Donner à l’utilisateur la possibilité de pause, avance rapide, plein écran, HD ou SD, téléchargement des transcripts (plupart du temps sous-titrage)... (voir Illustration 1)
4. Emmission de “tracking logs” compatibles avec les standards Open edX Video Interaction Events

Illustration 1 :



Faire son lecteur vidéo : les exemples

Ces lecteurs existants peuvent servir de base à une implémentation ou une extension des fonctionalités du lecteur existant. Le lecteur natif Open edX n’est pas un Xblock et fait encore partie du coeur du code Open edX video_module.py, ce qui nécéssite pour modification de faire un “fork”, peu désirable.

Voici quelques exemples de lecteurs vidéos intégrés par des Xblocks :

— Un lecteur Video JS : https://github.com/MarCnu/videojsXBlock

— Le lecteur Paella Player : https://github.com/polimediaupv/paellaXBlock
— Le lecteur Ooyala : https://github.com/edx-solutions/xblock-ooyala
— Le lecteur Brightcove : https://github.com/edx-solutions/xblock-brightcove
— Le lecteur DM Cloud : Le lecteur DM Cloud

Utilisation des lecteurs natifs

Il est possible d'utiliser les lecteurs natifs des plateforme d'hébergement et de les intégrer dans un Xblock. Dailymotion, sponsor de cet événement, sera là pour vous guider quant à l'intégration de fonctionnalités proposées dans les projets (Quiz Vidéo...).

Voci quelques liens vers l'API dailymotion. - Lien vers le site développeur https://developer.dailymotion.com/

Il existe un nouveau player HTML5 avec une API similiaire à l'ancienne API

Vous trouverez plus d'information ici sur le Nouveau Lecteur.

Daily Motion - Quelques informations supplémentaires

Voici un résumé court des fonctionalités du lecteur, modifiables par l' API.

Il est possible de :

— modifier les couleurs dans la barre de contrôle
— d'afficher ou non le logo dailymotion dans cette barre
— les partenaires "verified" peuvent appliquer leur logo en bas à droite des videos avec un lien vers leur site.

Le paramètre player qui permet de forcer le nouveau player : GK_PV5=1 dans l'API

Côté accessibilité, le player a été pensé pour répondre aux problématiques : possibilité de naviguer dans le player avec la touche "tab" ; il intègre les fonctionnalités dédiées aux non-voyants ("screen reader" et "voice over") ; les raccourcis clavier de sublime ont été conservés (documentation sur Sublime)

Administration

MySQL

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

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

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

```
fun lms.dev dbshell
```

MongoDB

Quelques commande pour acceder 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()
```

Apres un reboot de la machine virtuelle, il arrive que le service Mongo ne redemarre pas :

```
sudo rm /edx/var/mongo/mongodb/mongod.lock
sudo service mongodb start
```

ElasticSearch

Le service ElasticSearch écoute sur le port 9200 :

```
$ curl localhost:9200
{
  "ok" : true,
  "status" : 200,
  "name" : "Manbot",
  "version" : {
    "number" : "0.90.11",
```

```
    "build_hash" : "11da1bacf39cec400fd97581668acb2c5450516c",
    "build_timestamp" : "2014-02-03T15:27:39Z",
    "build_snapshot" : false,
    "lucene_version" : "4.6"
  },
  "tagline" : "You Know, for Search"
}
```

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
```

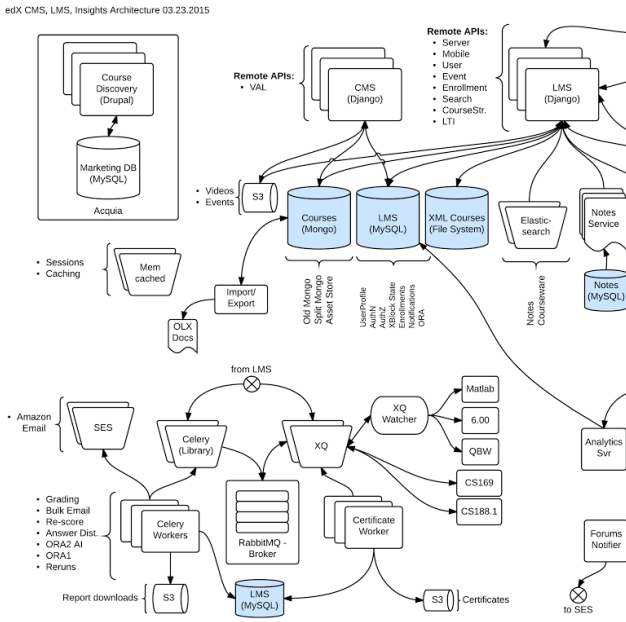
Les tracking logs qui agrègent le comportement des utilisateurs du lms :

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

Architecture générale

Comme précisé dans le document suivant Open edX Architecture, la plateforme se base notamment sur les technologies suivantes : - Python - Django - Mako Templates - CoffeeScript - Backbone.js - Sass et Bourbon - Ruby (Forum)

Sur le plan des composants : - Le LMS - Le studio - Les bases de données (MySQL pour les données étudiants et administration, Mongodb pour la structure de cours et données des forums)



Voici un schéma général d'une installation Open edX :