

# Login sur Stample

Ben Rajeb Moncef,  
Université de Jean Monnet

August 18, 2013

## Contents

<b>1</b>	<b>Intégration du module SecureSocial</b>	<b>2</b>
1.1	Introdutction . . . . .	2
1.2	Configuration du projet Stample . . . . .	2
<b>2</b>	<b>Schéma Model View Controller (MVC)</b>	<b>3</b>
2.1	Cycle de vie d'une requête . . . . .	4
<b>3</b>	<b>Présentation du contexte Business</b>	<b>6</b>
3.1	Login V0 . . . . .	6
3.2	Critiques . . . . .	6
3.3	Fonctionnalités manquantes . . . . .	6
3.3.1	Envoi des mails . . . . .	6
3.3.2	Authentification avec d'autres plateformes . . . . .	6
3.3.3	Rest mot de passe . . . . .	6
3.3.4	AuthCode . . . . .	7
<b>4</b>	<b>Première approche</b>	<b>7</b>
4.1	Etude du modèle . . . . .	7
4.2	Première Solution . . . . .	7
<b>5</b>	<b>Remise à Zéro</b>	<b>7</b>
5.1	Phase préliminaire . . . . .	7
5.2	Implémentation & Intégration . . . . .	8
5.3	Keep user Logged In SecureSocial :Authenticator Cookie . . . . .	8
5.4	Customisation des templates . . . . .	8
5.5	Partie Administrateur sur Stample . . . . .	9
<b>6</b>	<b>Conclusion</b>	<b>9</b>

# 1 Intégration du module SecureSocial

## 1.1 Introduction

Dans une première partie suite à la tâche "Login sur Stample" assignée dans *Trello*<sup>1</sup> avec Jonathan Winandy, j'ai commencé à lire, comprendre le code du backend Stample et à voir l'architecture du projet.

Problématique, Comment faire pour intégrer une couche horizontal sans tout casser coté Frontend ?

Au début, j'ai implémenté une solution basique, relativement fonctionnel mais non achevé suite aux complexités reliées à la compatibilité avec le code existant. Ensuite, remise à zéro en appliquant les méthodes d'ingénierie nous avons réussi à finaliser la tâche en moins de temps.

## 1.2 Configuration du projet Stample

J'ai eu l'accès au code source du projet privé sur le compte Git d'Edward avec son autorisation. Au début, j'ai utilisé un outil windows pour la gestion de mes projets Git d'apprentissage. Ensuite, j'ai changé mon PC parce que les outils SBT, play, mongo.... ont des besoins importants en ressources (mémoire,CPU). Stample m'a prêté un MacBookPro. Pour interagir avec Stample, j'ai consolidé mes connaissances dans le terminal-land (tmux<sup>2</sup>, zsh<sup>3</sup>, bash, git et SBT). Le README.md du projet écrit avec le langage de balises légers Markdown<sup>4</sup> contient les détails nécessaires pour la configuration du projet.

---

<sup>1</sup><https://trello.com/>

<sup>2</sup><http://tmux.sourceforge.net/>

<sup>3</sup><https://github.com/robbyrussell/oh-my-zsh>

<sup>4</sup><http://fr.wikipedia.org/wiki/Markdown>

## 2 Schéma Model View Controller (MVC)

En effet, dans la plateforme Stample il s'agit des requêtes affectées à des contrôleurs à l'aide de règles de routage, ces derniers impliquent des services dans un package services et sont implémentés dans un sous package "impl" (implementation) puis de même pour les repository et des views qui contiennent les templates. C'est une architecture de play et généralement un design classique d'une application java que j'ai découvert.

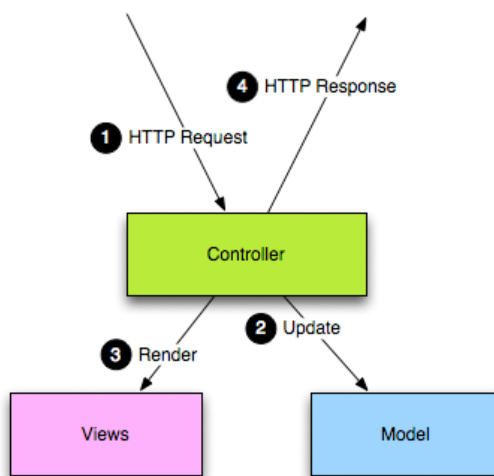


Figure 1: Diagramme MVC

- **Model** : Le modèle est la représentation spécifique au domaine de l'information sur laquelle l'application fonctionne. La logique de domaine ajoute «sens» aux données brutes (par exemple : le calcul des totaux, taxes de l'utilisateur, les frais d'expédition pour un panier ...). La plupart des applications utilisent un mécanisme de stockage persistant comme une base de données pour stocker des données. MVC ne mentionne pas spécifiquement la couche d'accès aux données, car il est entendu d'être en dessous, ou encapsulé par le modèle.
- **View** : La vue rend le modèle dans une forme appropriée pour les interactions, en général une interface utilisateur. Plusieurs views peuvent exister pour un modèle unique, à des fins différentes. Dans un format de préférence "Web" (HTML, XML ou JSON) en fonction de la négociation avec le navigateur et des capacités du contrôleur.

- Contrôleur : Le contrôleur répond aux événements (généralement des actions de l'utilisateur) et les traite, et peut également invoquer des changements sur le modèle. Dans une application Web, les événements sont généralement des requêtes HTTP: un contrôleur écoute les requêtes HTTP, extrait les données pertinentes de la «événement», telles que les paramètres de chaîne de requête, demander des têtes ... Et applique les modifications sur les objets du modèle sous-jacent.

*Dans une application play ces trois couches sont définies dans un répertoire app, chacun dans un package,*

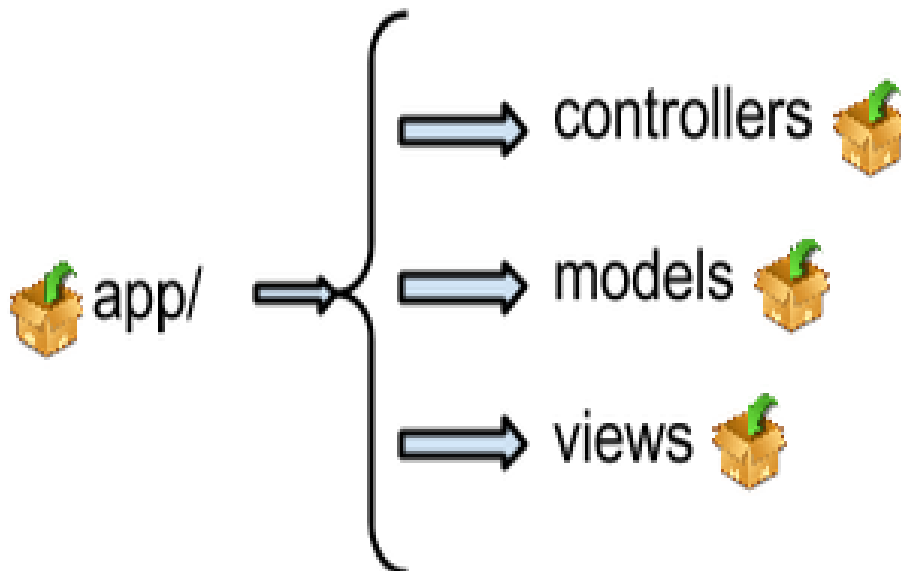


Figure 2: Default play packages

*ainsi que d'autres packages dans le projet Stampede comme event, api, services, repository, plugins...*

## 2.1 Cycle de vie d'une requête

Le framework play est entièrement Stateless[1] et orientée Request/Response. Tout les requête HTTP suivent le même path.

1. Une requête HTTP reçue par le framework.

2. Le Router Component essaie de trouver la route la plus spécifique en mesure d'accepter cette demande. Pour invoker la méthode d'action correspondante.
3. Le code d'application est exécutée
4. Si une vue complexe doit être générée, un fichier template est rendu.
5. Le résultat de la méthode d'action (Code de réponse HTTP, Content) s'écrit alors comme une réponse HTTP.

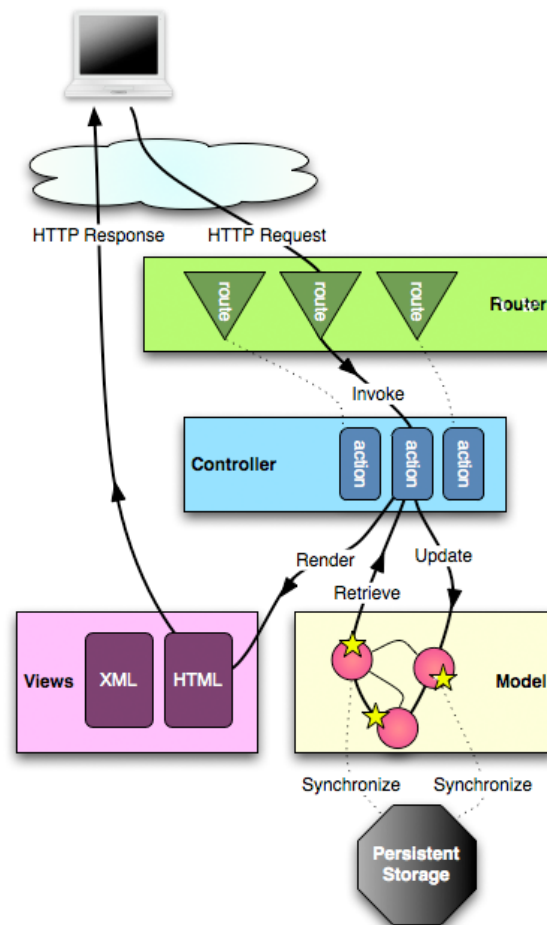


Figure 3: Diagramme HTTP request path

## **3 Présentation du contexte Business**

### **3.1 Login V0**

Dans la version précédente, il s'agit d'un formulaire classique à remplir pour créer un nouveau compte utilisateur qui sera stocker avec les coordonnées nécessaires (firstname, fullname, username, password, secretCode). Cela est une première approche, comme la plateforme est en cours de construction des nouveaux besoins apparaissent au fur et à mesure. Pour plusieurs raisons : Satisfaire la clientèle (utilisateurs de Stample), faciliter la perception de l'interaction et offrir des nouvelles fonctionnalités classique mais indispensable.

### **3.2 Critiques**

Cette approche pour la gestion de la création d'un compte utilisateur sur Stample doit être refondue. Il nous manque l'envoi des emails d'information lors des différents étapes d'inscription notamment le reset du mot de passe, ainsi que la possibilité de créer un compte à partir d'une autres plateformes en mesure que l'utilisateur s'ennui de reprendre tout un formulaire pour créer un nouveau compte sur une nouvelle plateforme.

### **3.3 Fonctionnalités manquantes**

#### **3.3.1 Envoi des mails**

Les utilisateurs de la plateforme doivent être notifier par mail lors de l'inscription, l'activation du compte et le changement du mot de passe. Les différents emails nécessaires sont signupEmail, welcomeEmail, alreadyRegisteredEmail, unknownNotice, passwordResetEmail, passwordChangeNotice.

#### **3.3.2 Authentification avec d'autres plateformes**

Plusieurs sites web offrent à leurs utilisateurs la possibilité d'utiliser gmail, facebook, twitter ou autres pour le signup. Cette fonctionnalités facilite l'inscription et rendre cette tâche rapide pour certains utilisateurs.

#### **3.3.3 Rest mot de passe**

Plusieurs utilisateurs, changent leurs mot de passe d'un site à un autre ce qui provoque souvent l'oubli du password. La gestion des mots de passe oublié est parmi les fonctionnalités indispensable sur une plateforme.

### 3.3.4 AuthCode

La version Beta de Stample est protégée avec un code d'activation, c'est une pratique habituelle pour sécuriser une plateforme mise en ligne et en cours de construction.

## 4 Première approche

### 4.1 Etude du modèle

Après quelques jours pour faire des considérations de conceptions et de mieux comprendre SecureSocial, j'ai réalisé que la mise en œuvre des méthodes n'était pas difficile à comprendre. C'est bien la conception de la logique dans un service backend qui compte. SecureSocial offre des APIs Scala et Java, on utilise ce module Scala pour le Backend de la plateforme. Les services d'authentifications utilisent la catégorie des services de premiers plans comme Google, Twitter, Facebook ect..., Il offre aussi un mécanisme de username/password avec les fonctionnalités Signup, Login, Rest Password. Ce module est compatible avec les versions de Play 2.1.x, 2.0.x et 1.x, il est relativement rapide à intégrer dans une application en suivant les instructions sur *le site*<sup>5</sup>. SecureSocial est extensible, il est basé sur un modèle de design qui vous permet d'ajouter des nouveaux services d'authentifications.

### 4.2 Première Solution

Les étapes d'implémentation:

- J'ai ajouté une nouvelle table dans la base de donnée Stample pour mettre le hash (email, random number, time ),
- Envoyer le mail avec un lien `url?hash=$hash`,
- Enfin vérifier l'existence du hash dans la base puis faire le tri.

## 5 Remise à Zéro

### 5.1 Phase préliminaire

D'après Jonathan Winandy, pour réussir il faut passer par les étapes suivantes :

---

<sup>5</sup><http://securesocial.ws/guide/getting-started.html>

- Do It :Commencer par poser le problème et puis écrire du code qui répond a ce besoin.
- Do It-Right : Se débarrasser du code inutile, faire des testes et des amélioration.
- Do It-Fast : Cleaner le code et vérifier les tests.

## 5.2 Implémentation & Intégration

Pour l'intégration de SecureSocial

$$DoIt = \begin{cases} \text{La vérification de compilation, intégration basique de secure Social dans Stample.} \\ \text{Une écriture basique dans la mémoire et l'implémentation de UserService.} \\ \text{Working Wiring : Authentification avec email.} \end{cases}$$

$$DoIt - Righ = \begin{cases} \text{Ajout du template signUp email.} \\ \text{Résoudre les problèmes de Token et de Memory.} \end{cases}$$

*L'étape Do It-Fast n'est pas encore faite mais elle pourrait être mise dans le planing des tâches plus tard.*

## 5.3 Keep user Logged In SecureSocial :Authenticator Cookie

Dans le fichier de configuration de SecureSocial il y avait des changements à faire comme la configuration des cookies comme absoluteTimeoutInMinutes(La durée d'authentification d'un utilisateur dans une session. Après cela l'utilisateur doit relogger (par défaut à 720 minutes - 12 heures). idleTimeoutInMinutes(La durée de session valable depuis la dernière requête (par défaut 30)). Comme le sujet, l'idée est de limiter les partie amovible, donc on s'est simuler une implimentation in-memory de l'authentificaion pour commencer.

## 5.4 Customisation des templates

Après l'intégration et la stabilisation des différentes interractions avec le module, j'ai travaillé pour refactoriser les différentes views pour s'adapter avec ce module (le signup, resetPasswordPage, authorisationCode, startResetPassword) et un main global. L'activationCode c'est un code secret pour protéger la version Beta de Stample(autoriser l'accé qu'à certain utilisateur),



il y avait un `activationCode` écrit dans le code de la plateforme que nous avons enlevé pour mettre dans l'interface admin la possibilité de générer des différents codes d'autorisations.

## 5.5 Partie Administrateur sur Stample

Stample c'est un réseau social en cours de construction il y avait plusieurs bugs et amélioration à ajouter sur la plateforme durant mon stage. Parmi ces améliorations l'ajout dans l'interface administrateur de l'API la possibilité de générer/écrire un code secret pour les utilisateurs lors de l'inscription avec un nombre d'usage pour chaque code. Cela nécessite l'ajout des routes pour gérer et créer les codes d'authentification dans l'admin controller et les templates. J'ai implémenté les différentes templates (`authorizationCodes`, `createCode`). Depuis l'interface Admin le bouton *create add autorization code*

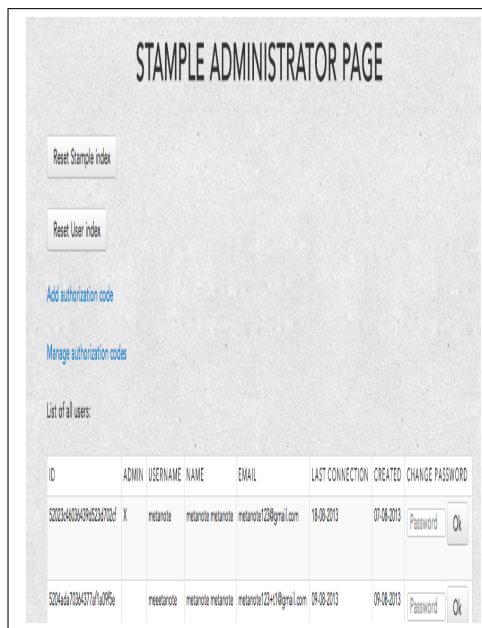


Figure 4: Admin page

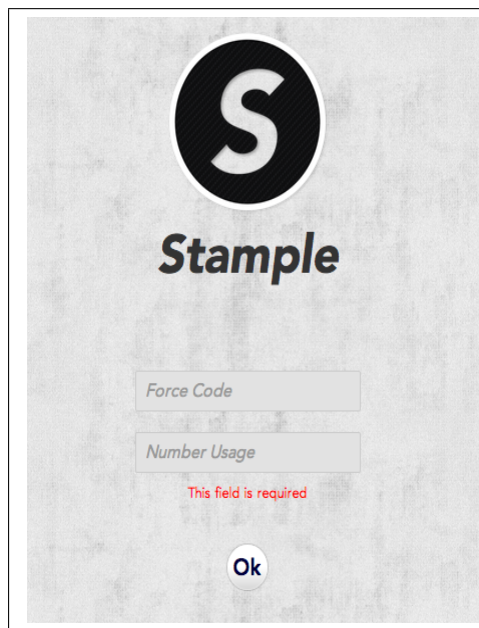


Figure 5: Create secret interface

ouvre la deuxième interface pour la création d'un code secret aléatoire avec un le nombre d'usage qui est un champ obligatoire ou bien en choisissant un code d'autorisation. Le bouton *manage autorisation codes* permet d'afficher les codes d'autorisation enregistrer et le nombre d'usage restant.

## 6 Conclusion

Cette partie d'authentification sur Stample m'a beaucoup apporté en écrivant du code/des templates et en fixant des bugs. Sur Backend, l'interaction des controllers, les models et les views avec les différentes classes et traits qui existant déjà m'avait beaucoup aider à comprendre le langage en ajoutant du code compatible avec ce qui existe. Cette partie du développement m'a permis de reconnaître le plugin SecureSocial qui peut servir dans d'autres projets et d'améliorer mes connaissances. Ce plugin sera peut-être enlever plus tard et remplacer par l'authentification avec WebID.