# Notification & Commentaire

### Ben Rajeb Moncef, Université de Jean Monnet

### August 20, 2013

## Contents

1	Introdutction	2
2	Notifications	2
3	Commentaires	2
4	Conclusion	4

#### 1 Introdutction

Dans une autre phase d'implémentation de la plateforme, comme autres réseaux sociaux il est indisponsable de notifier l'utilisateur dans le cas du partage d'un stample, une catégorie, le demande d'ajout...

Stample offre la possibiliter d'organiser le contenu sous forme des catégories, le partage de de ces dérnier c'est une nouvelle fonctionnalité qu'on trouve sur dropbox, sauf que pour Stample les catégories sont organisées sous forme d'une arborescence similaire à un système de fichier.

Le stample ou autrement l'article c'est un contenu qui peut être écrit en précisant le titre, le contenu, l'auteur, la source.... Peut être aussi générer par des darg & drop en utilisant le clipper depuis un autre site.

Ces options nécessitent d'informer les utilisateurs de la plateforme en option avec l'email et sous forme d'une alert sur le site.

La discution ou les commentaires sur un stample est aussi une fonctionnalité interessante pour interragir avec leur contenu. Dans cette partie j'ai travaillé sur des sous tâches pour les Notifications, et des tests d'intégrations pour les commentaires.

#### 2 Notifications

Le systéme de notification représente dans la base mongo dans une collection notification\_inbox cette dernière elle se compose d'un ID, notification et isRead. On utilise des servises : addNotificationsForUser, getNotification-ForUser, deleteNotification, readNotification.

#### 3 Commentaires

Pour partie des commentaires, j'ai eu des tests d'intégrations dans le commetServiceTest. Les tests sont écritent en Specs2 (software specification for Scala) : c'est une librérie pour écrire des spécifications des softwares executables et lancer sur la console avec sbt "it:test". Vous trouverez par suite deux captures d'écran une pour le lancement du test sbt et l'autre du code. Il faut ajouter les dependencies du plugin dans le fichier ApplicationBuilt du projet.

```
import models.{StampleComment, User}
               il authorizationCode.scala.html
                                                             import api.values.{StampleId, CommentContent, UserId, CommentId}
import api.commands.{UpdateComment, DeleteComment, CreateComment}
import controllers.ApiSpec
class CommentServiceTest extends Specification with BeforeAfterExample {
               main.scala.html
               i resetPasswordPage.scala.html
               signUp.scala.html
              startResetPassword.scala.html
           i fileUpload.scala.html
                                                                     TestHelper, cleanDatabase
           index scala html
                                                             protected def after: Any = ()
           indexClipping.scala.html
           login,scala,html
           loginClipping.scala.html
                                                                  protected def service = StampleTestApplication.get.defaultCommentService
           notFound.scala.html
                                                              □ "I should comment in my Stample" should {
       O Global
▼ 🗀 conf
       application.conf
                                                                     val command = CreateComment(StampleId(),UserId(),CommentContent("Comment Content"))
       logger.xml
       messages
                                                                     "I get a new comment" in {
       play.plugins
                                                                         val result : ApiSpec.ApiError[StampleComment] = service.process(command)
       routes
                                                                        result.isRight must beTrue
       securesocial.conf
                                                                        val comment = result.right.get
comment.author must be_==(command.asUser)
▶ □ logs
▼ 🛅 project
                                                             À
   ▼ 🗀 project
                                                                     "I should delete this comment" in {
    val comment=CreateComment(StampleId(),UserId(),CommentContent("comment to delete"))
    val results: ApiSpec.ApiError[StampleComment] = service.process(comment)
                                                             0
      ▼ 🛅 target
          ▶ ☐ config-classes
   ▼ 🗀 target
                                                                        val a = results.right.get
val command = DeleteComment(a.commentId, a.author)
      ▶ ☐ resolution-cache
                                                                        command.userId.id <u>must</u> be_==(comment.asUser.id)
       ▶ ☐ streams
                                                                     "I should update this comment" in {
   val comment = (CreateComment(StampleId(), UserId(), CommentContent("comment non updated")))
   val results: ApiSpec.ApiError[StampleComment] = service.process(comment)
       a build properties
    ▼ @ Build.scala

    ApplicationBuild

                                                                        val a = results.right.get
       plugins.sbt
                                                                        a.content.content must be_==("comment non updated")
▼ 🗀 public
                                                                        val command = UpdateComment(a.commentId, a.author, CommentContent("comment updated"))
   ▶ □ build
                                                                        command.content.content.toString must be_==("comment updated")
   ▶ □ css
```

Figure 1: Tests code

```
[info]
[info] Total for test service.CategoryServiceTest
[info] Finished in 0.618 seconds
[info] 8 tests, 0 failures, 0 errors
[info] service.CommentServiceTest
All collections have been cleaned and the database is empty
All collections have been cleaned and the database is empty
All collections have been cleaned and the database is empty
[info] CommentServiceTest
[info]
[info] I should comment in my Stample should
[info] + I get a new comment
[info] + I should delete this comment
[info] + I should update this comment
[info]
[info]
[info] Total for specification CommentServiceTest
[info] Finished in 111 ms
[info] 3 examples, 0 failure, 0 error
[info]
[info] + 'UserId(521291f703640e9cf6e37a24)' is equal to 'UserId(521291f703640e9cf6e37a24)'
[info] + '521291f703640e9cf6e37a27' is equal to '521291f703640e9cf6e37a27'
[info] + 'comment updated' is equal to 'comment updated'
[info]
[info]
[info] Total for test service.CommentServiceTest
[info] Finished in 0.149 seconds
[info] 3 tests, 0 failures, 0 errors
[info] service.StampleServiceTest
```

3

Figure 2: Terminal lance tests

### 4 Conclusion

Dans cette partie du Stage, j'ai découvet le systéme de notification en travaillant sur des sous tâches, une première approche pour le commentStample et j'ai écris les tests nécessaires pour la creation, la mise à jour et la suppression des commentaires. Le chapitre postérieur sera des altérnatives sur les reséaux sociaux centralisées et distribuées relative un future Stample en cours de construction. Des bibliothéques sont misent en ligne un projet open source "banana-rdf" et "RWW-Play", Henry Story, travaille sur ces bibliothéques pour les adaptées au plateforme Stample.