Couchbase l’utilise :

<http://fr.slideshare.net/Couchbase/reactive-programmingrxjavaefficientdata-benchristensenmichaelnitschinger?related=2>

<http://blog.couchbase.com/why-couchbase-chose-rxjava-new-java-sdk>

Expliquer en premier l’aspect fonctionnel : Par exemple filtrer avec un foreach puis un filter ()

Ensuite expliquer l’aspect reactif : on a la donnée maintenant vs on a la donnée plus tard

Faire un exemple sur l’affichage de données sur un histogramme polaire

Exemple de présentation : <http://fr.slideshare.net/samhogy/meet-rxjava-your-new-best-friend>

Site général : <http://reactivex.io/>

Tutorial interactif directement dans le navigateur : <http://jhusain.github.io/learnrx/>

1. Bref historique
   1. Né avec Microsoft
   2. Porté en Java par Netflix
   3. Utilisé par d’autres boites connues : CouchBase, etc.
2. Qu’est ce que c’est ? Reactive Functionnal Programming
3. Reactive…
   1. Tableau from <http://reactivex.io/intro.html> (1er tableau)
      1. Il manque la case du bas droite
      2. C’est un observable. Observable ?
   2. « Book design pattern »
      1. Pattern Observer
      2. Observable 🡪 Observer : onNext
      3. 2 ajouts pour être équivalent à un itérateur from <http://reactivex.io/intro.html> (2ème tableau)
         1. onCompleted
         2. onError
4. … Functionnal : + plein de fonction pour manipuler les données
5. Quelques fonctions courantes
   1. Foreach (PPT1)
   2. Map (PPT1)
   3. Filter (PPT1)
   4. ConcatAll (PPT1)
6. Mettre un écran « fin » façon cinéma muet à la fin de la courte explication pour montrer que c’est court
7. Exemple 1 : D’une présentation de Netflix, récupérer les films les mieux notés pour les suggérer
   1. PPT1 p.22
8. Exemple 2 : Recherche instantanée
   1. PPT1 p.66
   2. On commence à écouter les évènements utilisateurs et à les brancher à la fonction de recherche (fonction interne ou webservice)
   3. Inutile de lancer une recherche à chaque caractère : throttle
   4. Encore plus faineant : si l’utilisateur a tapé une lettre et l’a supprimé, on ne relance pas la requête : dinstinctUntilChanged
9. Exemple 3 : suite de l’exemple 2
   1. Qu’est ce qu’on fait du résultat de la recherche ? On le met dans l’interface
   2. Problème : tout ca est sur le thread UI et ca rame !
   3. Solution : On passe tout en multithread… en deux méthodes
   4. [Facultatif] si on tape deux lettres, comment la première requête sait qu’elle doit s’arrêter ?
      1. L’observer qui récupère les éléments résultats s’arrête dès qu’une nouvelle entrée est saisie : searchResults.takeUntil(unserInput)
10. Grosse démo (Appli android facultative)
    1. Schéma block global
    2. [C++] PC qui envoie des données aléatoire à une fréquence de XXX
       1. Sur un Websocket du PC de démo (ZeroMQ)
       2. Sur le port bluetooth
    3. [JS] L’appli HTML du PC de démo récupère les données
    4. [JAVA] L’appli Android récupère les données
    5. L’appli Android peut aussi diminuer ou augmenter la fréquence d’échantillonnage de la source
    6. La plupart des blocs est réutilisable
       1. ZeroMQ
       2. Affichage Graph
       3. WebService
11. Comment l’utiliser
12. Bilan
    1. Un concept simple mais puissant
    2. Léger et facile à intégrer dans le projet
    3. Simplifie la vie : multithread, multiplateforme, UI
    4. Réutilisable
13. Bonus : Performance
    1. Benchmark multithread
    2. Benchmark avec Fibers
14. Liens
    1. Liens officiels
    2. Tutorial interactif directement dans le navigateur : <http://jhusain.github.io/learnrx/>
    3. <http://rxmarbles.com/> : Diagramme interactif pour comprendre comment ca marche