Autour de JavaFX

JavaFX sur plateforme ARM

Android et iOS comme cibles?

- Deux techniques répandues
  - JavaFX Android Ports
  - RoboVM iOS
- Pas de support officiel par Oracle
- Dans les deux cas: le livrable est un package natif
  - Pas de JVM Android ou iOS
  - Englobe la partie toolkit JavaFX
  - Le livrable est donc plus volumineux qu'une application native "classique"

JavaFX sur Android

OpenJFX porté sur Dalvik

- Prérequis
  - Android SDK
  - JavaFX-Dalvik runtime (port de OpenJFX)
  - Votre application JavaFX compilée en Java 7 uniquement
  - Gradle pour builder le projet Android
  - Ant pour créer le livrable Android
- Bilan
  - Non support de Java 8 dans son intégralité
  - Possibilité d'accéder aux fonctions natives d'Android depuis l'application JavaFX
  - Promesse de réutilisation de code entre Desktop et Android tenue!

JavaFX sur iOS

RoboVM à la rescousse

- Prérequis
  - Votre application JavaFX compilée en Java 7 ou 6
  - Une classe "launcher" spécifique
  - xcode pour traduire le bytecode Java en natif ARM
  - RoboVM runtime et cocoatouch dans les dépendances du projet
  - Maven pour construire le livrable iOS via robovm-maven plugin
- RoboVM utilise des bindings spéciaux "Java to Objective-C"
  - Support de Java 8 (mais pas de JavaFX 8 fourni par OpenJFX)
  - Code natif produit optimisé
  - Utilise LLVM comme compilateur

JavaFX sur iOS

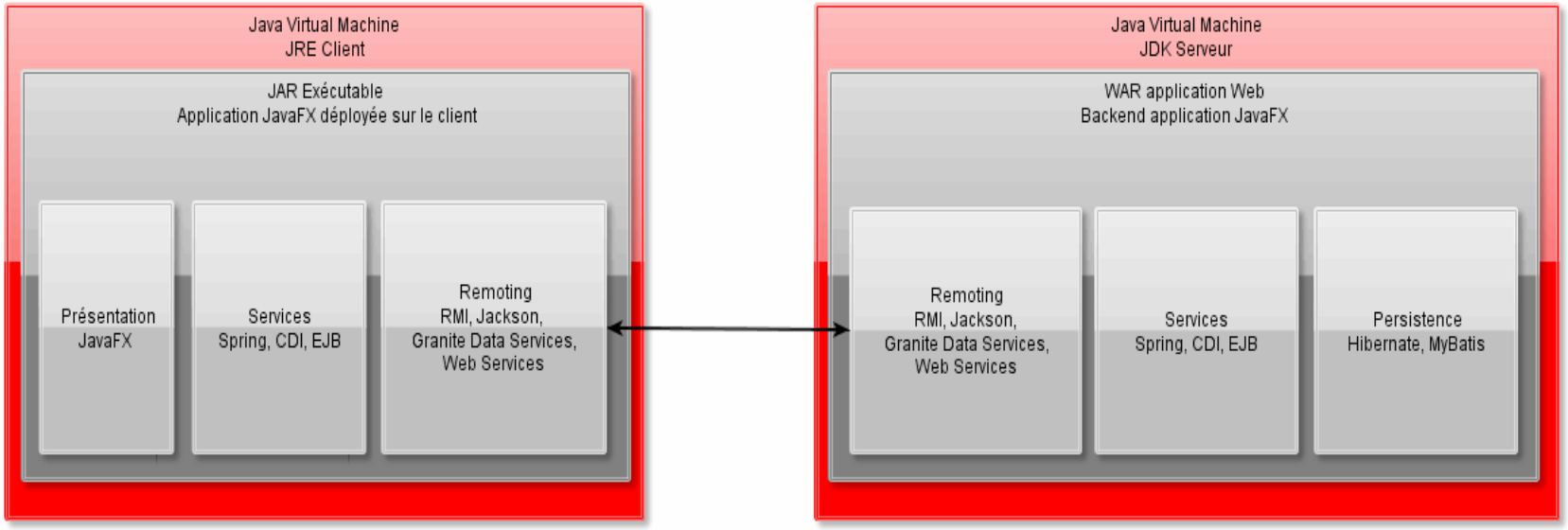
Conclusions

- Avantages
  - Support de RoboVM: un acteur important
  - Réutilisation de code
  - Possibilité d'appeler des fonctions natives
- Problématiques spécifiques
  - Appeler des fonctions natives est long
  - Complexifie le build
  - Il est aussi possible d'utiliser RoboVM pour faire des UI natives: JavaFX est-il pertinent sur une cible iOS / Android uniquement?

Architecture N-tiers

Schéma

Il peut être intéressant d'avoir un framework pour faciliter les synchronisations avec un serveur distant des données dans un mode distribué (comme Granite)



Granite Data Services

Framework complet orienté JEE

- Framework d'intégration serveur pour JavaFX, Android et Flex
- Permet de faciliter les échanges avec un serveur Java
  - Spring + Hibernate côté serveur ou EJB3
  - Sérialisation des entités vers le client JavaFX
  - Messaging via WebSockets compatible JMS
  - Génération de classes avec Property (compatibles databinding)
- Version simple gratuite open source
- Version payante complète open source

Granite Data Services

Résumé

- Avantages de la solution
  - Payload très compact (marshalling JMF ou AMF)
  - Possibilités d'architecture real-time
  - Multi-supports
  - Intégration poussée des solutions JEE / Spring / Hibernate
  - Intégration facile à une usine logicielle
  - Framework MVP côté JavaFX

- L'utilisation de Granite est très profitable si
  - Mode connecté >> Mode déconnecté
  - Stack technique Java importante
  - Le paiement de la licence Granite est possible

LANGAGES ALTERNATIFS

De nombreux choix

- GroovyFX
  - Utilisé à la place de FXML en mode compilé
  - Intégration possible avec Griffon (Grails pour frontend JavaFX)
- ScalaFX
  - Utilisé à la place de FXML en mode compilé
- FXGraph + XTend
  - Intégration dans E(fx)clipse native
  - FXGraph remplace FXML en mode interprété
  - Compilation possible de FXML / FXGraph (plugin instable en alpha)