Concurrency in Enterprise Java

Date : 9 avril 2015
Format : Conférence
Speakers : Arun Gupta, Red Hat

Avant de rejoindre Red Hat, Arun a passé 14 ans chez Sun / Oracle sur la stack JEE.

Rappel sur les threads en Java SE : interface Runnable et classe Thread.
Dans JEE, créé un nouveaux thread est considéré comme dangereux :

- Pas de gestion du cycle de vie des threads
- Pas de gestion du nombre de threads
- …

Pattern of software possibility :

- Timeouts : provide fallbacks
- Bulkhead : separate your application components
- Circuit breaker : échouer rapidement si aucun thread n'est disponible
- Strady State : libérer les ressources inutilisées

La JSR 256 :

- API standardisée pour utiliser la concurrence dans des applications d'Entreprise
- Extension du java.util.concurrent (JSR 166y) de Java SE
- Gestion et supervision apportée par le conteneur JEE

4 types d'objets managés font partis du package javax.enterprise.concurrent.

Qu'est-ce qu'un Managed Object ?

- Fournit par le conteneur JEE
- Récupérer via un lookup JNDI ou injecter avec @Resource
- Configurable (la spéc ne précise rien sur leur configuration)

- Objets par défaut pré-configurés dans Wildfly et Glassfish

Context and Notification

- Container context propagation
- Transaction management
- Task event notification

Les contextes sont propagés d'un Managed Object à l'autre. Chaque application déployée dans le même serveur JEE possède son propre contexte (ex : JNDI namespace, class loading).

Managed Object #1 ManagedThreadFactory

- Supporte la propagation de contexte
- Peut être utilisé avec les API asynchrones des servlets
- Etebd java.util.concurrent.ThreadFactory
- Le contexte est capturé à la création du ManagedThreadFactory, par exemple lors de l'injection ou du lookup JNDI. Et pas à la création du thread.
- Cette fabrique retourne des threads implémentant ManagedThread
- Peut être utilisé avec l'API Java SE concurrency utilities

Managed Object #2 ManagedExecutorService

- Les tâches doivent impléménter Callable ou Runnable
- Supporte la propagation de contexte
- Les patterns JavaSE s'appliquent
- Hérite de java.util.concurrent.ExecutorService
- Un Future est retourné une fois la tâche soumise.
- Attention : les méthodes shutdown, shutwownNow … sont désactivés en JEE car c'est le conteneur qui gère le cycle de vie
- Utilisable dans les servlets asynchrone @WebServlet(asyncSuported=true)

Arun nous montre en live que WildFly permet de créer une ressource en ligne de commande. Wildfly propose également une web app pour générer des lignes de commandes CLI. Pratique pour l'ajouter dans un docker file pour créer les ressources JMS, JDBC …

Managed Object #3 ManagedScheduledExecutorService

- Permet de planifier une tâche après un certain délai, de manière périodique ou en fonction d'une expression cron
- Etend la classe ScheduledExecutorService de JavaSE
- Personnalisation possible à l'aide de Trigger : à implémenter par le développeur
- Lifecycle API's disabled

Managed Object #4 ContextProxy

- Quelque chose de nouveau par rapport à Java SE
- Permet de capturer le contexte pour l'utilisé ultérieurement

- Le contexte est capturé à la création
- Cas d'utilisation : propager l'identité d'une personne
- Personnalisable (transaction property, vendor specific properties)

Contexte transactionnel

- Le contexte transactionel n'est pas propagé.
- Mais le UserTransaction de JTA est disponible.
- Peut utiliser le même contexte transactionnel que le thread appelant

Task Events Notifications

- Assuré par ManagedTaskListener
- Permet de connaître l'état des tâches managées

Ressources :

- Plus de 200 exemples sur https://github.com/javaee-samples/javaee7-samples
- Java EE 7 course on Parleys