Stage pratique de 4 jour(s)
Réf : JQT

Participants

Développeurs, architectes logiciels et chefs de projets.

Pré-requis

Connaissances de base en Java et de l'utilisation d'un IDE (Eclipse, IntelliJ...).

Dates des sessions

CLASSE A DISTANCE
15 juin 2021, 14 sep. 2021

Modalités d'évaluation

L'évaluation des acquis se fait tout au long de la session au travers des multiples exercices à réaliser (50 à 70% du temps).

Compétences du formateur

Les experts qui animent la formation sont des spécialistes des matières abordées. Ils ont été validés par nos équipes pédagogiques tant sur le plan des connaissances métiers que sur celui de la pédagogie, et ce pour chaque cours qu'ils enseignent. Ils ont au minimum cinq à dix années d'expérience dans leur domaine et occupent ou ont occupé des postes à responsabilité en entreprise.

Moyens pédagogiques et techniques

• Les moyens pédagogiques et les méthodes d'enseignement utilisés sont principalement : aides audiovisuelles, documentation et support de cours, exercices pratiques d'application et corrigés des exercices pour les stages pratiques, études de cas ou présentation de cas réels pour les séminaires de formation.

• A l'issue de chaque stage ou séminaire, ORSYS fournit aux participants un questionnaire d'évaluation du cours qui est ensuite analysé par nos équipes pédagogiques.

• Une feuille d'émargement par demi-journée de présence est fournie en fin de formation ainsi qu'une attestation de fin de formation si le stagiaire a

Java, programmation multithread

La programmation multithread en Java prend de plus en plus d'importance grâce à la généralisation des architectures multiprocesseurs : elle simplifie la conception et le développement des applications comportant un parallélisme intrinsèque et offre des solutions efficaces aux problèmes de performance.

OBJECTIFS PEDAGOGIQUES

Maîtriser les modèles de programmation multithread et les librairies standards correspondantes
Connaître les principales structures de données adaptées à la programmation multithread
Connaître les principaux bugs et avoir un aperçu des solutions
Connaître les outils de test et débogage
Comprendre les liens entre la programmation multithread et les performances

1) Les modèles, les interfaces et classes de programmation multithread

- Les concepts de base : tâche, ressource d'exécution, activité, service d'exécution, futur.
- Les différentes mises en œuvre des concepts (Runnable, Callable<T>, ExecutorService, Future<T>…).
- Les exceptions non catchées, les groupes de threads.
- Le futur complétable.

Travaux pratiques
Programmation d'une application combinant les différents modèles.

2) Les contraintes sur le comportement correct des activités

- Quelques méthodes formelles de spécifications.
- Une méthode semi-formelle de spécification.
- La mise en œuvre des spécifications.

Travaux pratiques
Utilisation de méthodes formelles de spécifications.

3) La synchronisation et communication des threads

- Les status "synchronized", "wait", "notify" et la programmation de moniteurs.
- Les interfaces et classes de synchronisation : verrous, sémaphores, les barrières cycliques.
- Les queue.

Travaux pratiques
Utilisation des interfaces et classes de synchronisation.

4) L'exécution de tâches en parallèle

- Les ExecutorService.
- Le modèle fork/join (RecursiveTask<T>, RecursiveAction, ForkJoinPool).

Travaux pratiques
Utilisation des services d'exécution et du modèle fork/join.

5) Les structures de données dédiées à la programmation multithread

- Les collections spécialisées.
- Le stockage local des données de thread : ThreadLocal<T>.
- Les classes Atomic.

Travaux pratiques
Utilisation des structures de données.

6) Les threads et les performances

- L'impact de la création de threads.
- L'impact de la synchronisation.
- L'impact des caches mémoire.
- Les threads et les IO, les BD et le graphique.
- Les threads et l'ordonnancement.

Travaux pratiques

bien assisté à la totalité de la session.

Optimisation des programmes.

7) Les modèles alternatifs

- Les modèles asynchrones : JReact.
- Les modèles d'acteurs (Akka Actor4J...).
- Les modèles réactifs synchrones.

8) Outils dédiés au développement de programmes concurrents

- JConsole, jstack.
- La librairie JArmus.
- La logique temporelle de Lamport : TLA+.
- Les erreurs courantes : famine (contention), endormissement (dormancy), interblocage (deadlock), terminaison prématurée.