

Bästa praxis inom Java-programmering - 4 dagar

kurser 516

- Du får lära dig att**
- Använda bästa praxis inom Java för att öka produktiviteten och skapa snabba, säkra och pålitliga applikationer
 - Automatisera driftsättande, testning och buggsökning i programvaruapplikationer
 - Lösa utvecklingsproblem med beprövade designmönster och avancerade språkfunktioner
 - Maximera programvarans prestanda
 - Förbättra trådade applikationers pålitlighet
 - Koda säkert i Java och autentisera med branschens säkerhetsramverk
- Sammanfattning** Inom Java finns många funktioner för att skapa kraftfulla, säkra och lättillgängliga applikationer. Enbart kunskap om språket och API räcker inte för att utnyttja kraften i Java fullt ut. Utvecklare måste utnyttja beprövade metoder och industristandarder vid programvaruutveckling. Kursen ger de färdigheter som behövs för att utveckla mjukvara och leverera driftsäkra applikationer.
- Vem bör delta** Utvecklare, arkitekter och alla som arbetar med Java-projekt och vill öka sina kunskaper inom Java-programmering. Erfarenhet av Java i nivå med kurs 471, "Java-programmering: omfattande introduktion", förutsätts.
- Praktiska övningar** Du tillämpar beprövade branschmetoder och får erfarenhet av att använda API och språkfunktioner. Övningarna omfattar:
- Förbättra testbarheten genom att skapa en tandemklass med dess enhetstest
 - Implementera viktiga objektorienterade designmönster för att lättare kunna utöka och underhålla systemet
 - Optimera programvarans prestanda genom att reorganisera slingor och minska databassamtal
 - Anropa dynamiska verksamhetsregler med skripting
 - Tillämpa säkerhetsrestriktioner

Bästa praxis inom Java-programmering - 4 dagar

kurser 516

Effektiv Java-programmering

- Tydliggöra målen för beprövade lösningar
- Identifiera egenskaperna hos högkvalitativ programvara

Optimera programvaruutvecklingen med beprövade metoder

Förenkla projektupbyggnad och driftsättande

- Automatisera utbyggnadsprocessen med Ant
- Kontrollera och konfigurera loggning

Tillämpa teststyrd utveckling

- Enhetstesta komplexa komponenter
- Sammanställa och underhålla JUnit-tester
- Automatisera "project-wide"-testning
- Validera applikationsresultat med funktionalitetstester
- Testa container-hanterade komponenter som servlets

Förbättra kodkvaliteten med bättre design

Expertrekommendationer

- Balansera utöknings- och underhållsmöjligheter
- Undvika problem med clone()
- Undantag i bästa praxis

Uppnå typsäkerhet

- Eliminera körtidsfel med generiska metoder
- Skriva generiska klasser och metoder
- Begränsa parametervärden med kanonisering

Framtvinga inkapsling

- Erbjud grovkorniga metoder med Memento
- Förenkla anpassning till gränssnitt

Refaktorisering och designmönster

- Effektivisera källkoden med refactoring
- Utforma gränssnitt för förbättrad programvaruflexibilitet
- Viktiga objekt-orienterade designmönster
- Template Method
- Strategy
- Composite
- Factory
- Inversion of control

Justera för maximal prestanda

Mäta prestanda

- Tillämpa verktyg för prestandaprofilering
- Bedöma svarstiden
- Utföra belastnings- och stresstester

- Identifiera flaskhalsar för prestandan

Strategier för att förbättra prestandan

- Metoder för att hantera vanliga frågor kring Javas prestanda
- Utnyttja genererande skräpinsamlare
- Välja lämpliga JVM- och containerinställningar
- Bedöma behovet av NIO och JNI
- Förbättra svarstiden med hjälp av reorganisering-slingor
- Bearbeta strömmande data för minskat minnesutnyttjande

Använda Collections API effektivt

- Förhindra minnesläckor med svaga referenser
- Välja bäst collection-klasser

Utnyttja trådar fullt ut

Förbättra svarstiden genom parallellisering

- Skriva pålitlig, trådsäker kod
- Undvika race hazards och deadlocks
- Använda ramverket executors

Skydda en trådad applikation

- Synkronisera trådar
- Metoder för datadelning mellan trådar
- Hantera prestandaimplikationerna med synkronisering

Tillämpa säkerhetsrestriktioner

Skydda applikationer

- Säker kodning i Java
- Begränsa åtkomsten till skyddade resurser
- Fastställa riktlinjer för säkerheten

Autentisering och auktorisation

- Tillämpa rollbaserad säkerhet
- Autentisera användare i webbapplikationer

Hantera förändringar med designmönster

Begränsa förändringens påverkan

- Centralisera egenskaper med Singleton
- Lägga till transparent beteende med Proxy
- Wrappa externa bibliotek med Adapter

Moderna designmönster

- Invertera kontrollen (IoC) med "bean factories"
- Anpassa beteende med aspekter
- Lägga till skriptingfunktion i en applikation
- Skapa skript anpassat till slutanvändarens beteende