itkonekt

PRAGMATIC DOMAIN-DRIVEN DESIGN (DDD)

Two-day workshop with Victor Rentea

www.itkonekt.com

VICTOR RENTEA

// Java Champion, ex Lead Architect @IBM

https://twitter.com/VictorRentea

Victor is a Java Champion, and one of the top Technical Trainers, **having trained more than 1.5K developers in dozens of companies worldwide**.

Victor's talks are regularly top-rated at the largest international conferences in Europe: his live-coding sessions are lightning-fast but well crafted, full of enthusiasm, deep insights and take-away tips.
His passion is Simple Design, Refactoring, and Unit Testing, about which he regularly talks at top conferences. His personal commitment is to seed passion for writing clean, professional code.

About workshop

PRAGMATIC DOMAIN-DRIVEN DESIGN (DDD)

Domain-Driven Design is a battle-proven technique invented almost 20 years ago, that can significantly simplify the evolution and maintenance of software tackling a complex domain problem. DDD can be applied at two levels: strategic at the organization level (eg. to delineate microservices), as well as tactical in day-to-day coding. First, we will review the DDD philosophy and the key strategic-level decisions. But the main part of the workshop will revolve around refactoring a traditional codebase built around an anemic domain model, towards a more DDD-style code using a Rich Domain and Aggregates. Throughout our journey, a question will follow us: “How does that help me?”, asked from a skeptical/pragmatical perspective.

What will you learn?

By the end of the live online course, you'll understand:

- ✓ Domain-Driven Design philosophy and goals
- ✓ What is a Ubiquitous Language and why it's critical when tackling a complex domain
- ✓ The benefits and costs of a Rich Domain Model
- ✓ The core DDD patterns at both strategical and tactical level
- ✓ The principles underlying the Onion-, Hexagonal-, Ports-and-Adapters, and Clean- Architectures

What will you learn?

You'll be able to:

- ✔ Implement the key DDD Tactical Patterns including Application/Domain Services, Entity, Value Object, Aggregate, Domain Events...
- ✔ Use Aggregate design in real-life scenarios
- ✔ Use Domain Events as an alternative to orchestration
- ✔ Progressively refactor a traditional codebase following DDD principles (the so-called "Lite DDD" approach)
- ✔ Apply many techniques in any project (even not "officially" DDD)

This workshop is for you if:

- YOU ARE A MID-SENIOR (JAVA) DEVELOPER WHO WANTS TO UPGRADE HIS/HER DESIGN SKILLS
- YOU ARE AN ARCHITECT THAT REGULARLY WRITES OR REVIEWS CODE THAT WANTS TO CAREFULLY DESIGN A CRITICAL PROJECT
- YOU ARE A TECHNICAL LEAD OF A PROJECT USING OR STARTING TO USE DDD
- YOU ARE AN AVID LEARNER WHO WANTS TO UNDERSTAND THE SOFTWARE ARCHITECTURE USED FOR THE MOST COMPLEX DOMAINS IN THE WORLD

Agenda

(covered in non-linear style)

- ✓ **Introduction** (2 hours)
  - DDD: Philosophy, when to use/avoid, typical misconceptions
  - Ubiquitous Language, Bounded Context, and Context Map
  - Strategic Relationships between Bounded Contexts
  - Distillation of the Core Domain
- ✓ **Clean Architecture** (2 hours)
  - Layered Architecture: Presentation, Application, Domain, Infrastructure; relaxed variant
  - Adapter, Dependency Inversion Principle, and Anti-Corruption Layer
  - Application Services vs Domain Services, and an evolutionary design strategy
  - The Onion Architecture (aka Ports-and-Adapters, Hexagonal) + pragmatic variant
  - Separating Bounded Contexts in independent sub-modules

Agenda

(covered in non-linear style)

- ✓ **DDD Tactical Patterns** (4-5 hours)
  - Entity, Value Object, preserving the integrity of domain invariants
  - Aggregate as the consistency boundary for changes
  - Domain Events and Eventual Consistency; Event Sourcing principles (brief)
  - How to avoid an Anemic Domain Model
  - Repository, Factory (briefly): keep them pragmatic
  - Identifiers: ID Type (microtypes), Semantic ID, Immortal or Immutable data upon publishing ID

Agenda

(covered in non-linear style)

- ✓ **Separating Bounded Contexts in Modules** (1 hour) - an incremental technique
  - Identifying internal bounded contexts
  - Removing entity links
  - Refining Bounded Contexts interactions
  - Handling Transactions, Foreign Keys, and Replication
- ✓ **DDD implementation tricks for Java** (<= 1 hour):
  - Hibernate tricks: ID Type, persisting immutable Value Objects, preserving invariants in an @Entity
  - Spring tricks: @DomainEvents, Stereotype annotations, Aspects, Spring Data JPA Specifications
  - javax.validation best practices, custom validators, validation profiles

Agenda

(covered in non-linear style)

- ✔ **Wrap-up** (1 hour):
  - Starting your first DDD project: the right domain, preparation, reading guide, planning, people
  - Pitfalls of DDD: the DDD God Fallacy, overengineering, internal framework mania, Tech-tiger bias
  - "Light DDD": 10 Takeaways applicable in any non-DDD Project
  - More open debates

Workshop details

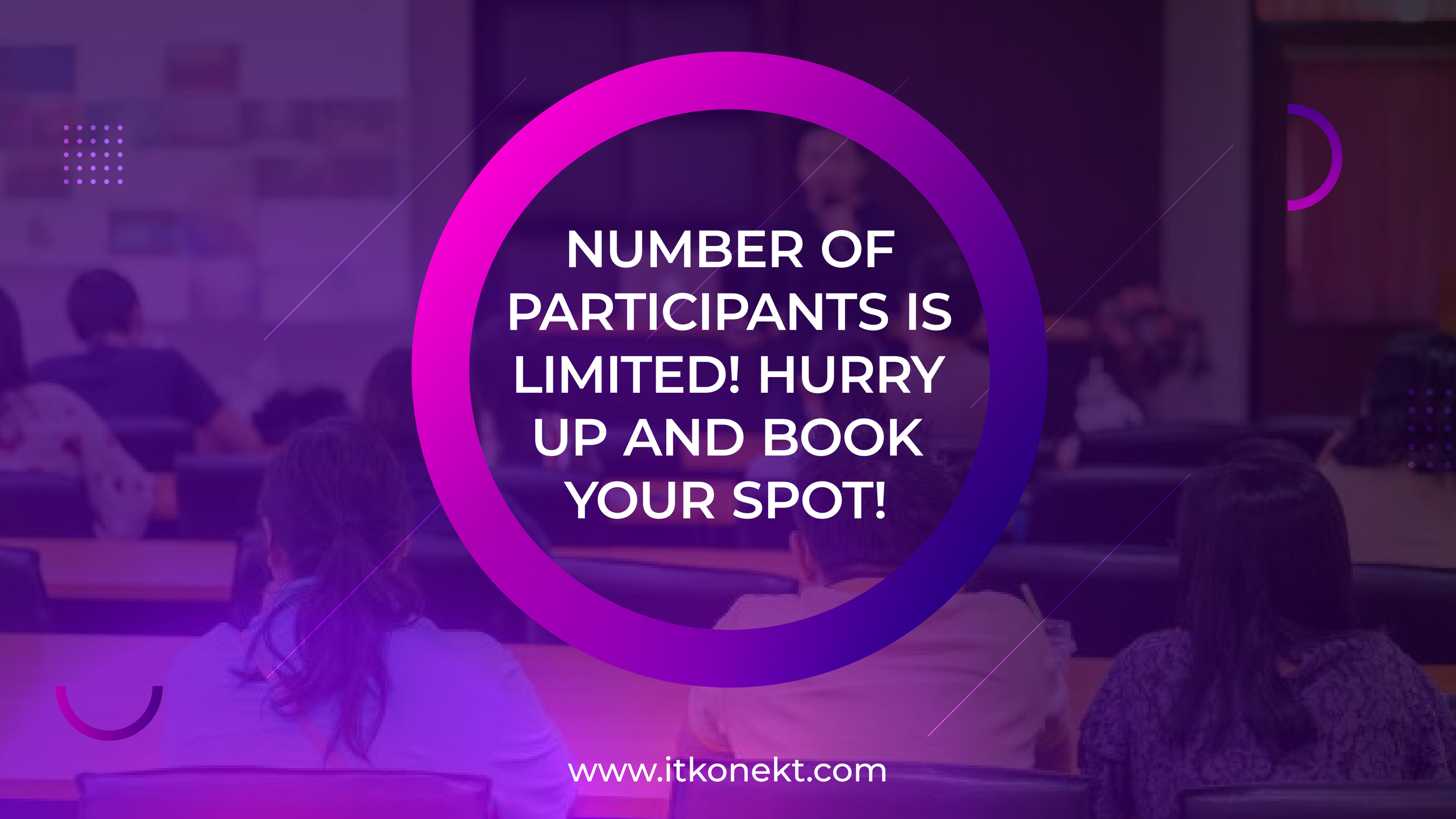
NOVEMBER
7TH & 8TH

TIME: 9h-17h CET

LIMITED NUMBER OF PARTICIPANTS: 25

ONLINE

LANGUAGE: ENGLISH


NUMBER OF PARTICIPANTS IS LIMITED! HURRY UP AND BOOK YOUR SPOT!
www.itkonekt.com