BO·TT·EGA
IT minds

Training program:

Domain Driven Design Implementation – architectural patterns (part 2)

Info:

Name:	**Domain Driven Design Implementation – architectural patterns (part 2)**
Code:	**DDD-impl**
Category:	DDD Workshop
Target audience:	developers
Duration:	2 days
Format:	50% lecture / 50% workshop

Pragmatic approach to the DDD implementation in selected technology: Java, NET, PHP, RoR.

Scope

Technical aspects of the DDD implementation

- application architecture: managing complexity thanks to separation of logic,
- system architecture: integration of modules, event architecture that increases responsiveness and opens for plug-ins,
- system scaling – CqRS architecture,
- specific techniques of implementation using popular technological stacks,
- best Clean Code practices and caring for high testability.

Business problems modeling techniques

DDD modeling techniques (strategic and tactical patterns as well as linguistic and visual techniques) are being talked over at the DDD-modeling training, which should be done first, before the training in the field of implementation.

Form

The training is based on a model created during the training in the field of modeling, which precedes this training.

During lectures, the coach talks over the best practices of implementing the application architecture patterns and DDD Building Blocks. During workshops, we implement two modules of the ERP class system. Next tasks consist of an incremental addition of new functionalities in a way that illustrates theoretical issues, learned during the lecture preceding them. During the discussions, participants have an access to the technical knowledge of the coach and have a possibility to verify their solutions with the ones developed by other participants of the training.

During workshops, participants solve the problems presented to them, working in pairs (Pair Programming), changing the roles after each task: Pilot and Driver. The purpose of this technique is to enable a look from different perspectives at the issues, at the same time activating more cognitive resources.

BOTTEGA IT minds

As a part of the training, we talk over and practice both basic and advanced DDD techniques, such as: Building Blocks patterns, developing the Ubiquitous Language and the Strategic Design technique set.

Reference project

Check out our implementation of an example DDD+CqRS project: Sample Leaven.

It's all about the content.

- Talking over the most common errors in the implementation of Aggregates
- Best practices and traps when using ORM and IoC
- Modern architectures: CqRS, Event Driven, Event Sourcing

BO·TT·EGA
IT minds

Training program

1. Application architecture – practical implementation of architectural concepts

- 1.1. Using the IoC when designing with quality in mind – design openness for extension and tests
  - 1.1.1. Dependency Injection – plug-ins that change the system behavior
  - 1.1.2. Events – plug-ins that add a new behavior
  - 1.1.3. AOP – orthogonal aspects (transactions, security, audits)
- 1.2. Layer approach - arranging building blocks on layers
  - 1.2.1. System structuring
    - 1.2.1.1. Modules – building
    - 1.2.1.2. Packets/namespaces
  - 1.2.2. Presentation layer
    - 1.2.2.1. Protecting the domain from leaking to the UI
  - 1.2.3. Application logic
    - 1.2.3.1. Designing the API of a system
    - 1.2.3.2. Problem of the fair API – object versioning
  - 1.2.4. Domain logic (Building Blocks DDD)
    - 1.2.4.1. Visibility of the classes – using the packet range
  - 1.2.5. Infrastructure layer
    - 1.2.5.1. ORM challenges
    - 1.2.5.2. Traps of the event architecture

2. System architecture

- 2.1. Microservices and SOA princuples
  - 2.1.1. The single source of truth rule
  - 2.1.2. Avoiding "Corporate Model" anti-patterns

BOTTEGA IT minds

BOTTEGA IT minds

BOTTEGA IT minds

- 3.7. Policies (strategies)
  - 3.7.1. Decorating
  - 3.7.2. Injecting policies as plug-ins
- 3.8. Business events
  - 3.8.1. Asynchronous approach
  - 3.8.2. Plug-in architecture
- 3.9. Specifications
  - 3.9.1. Modeling complex business conditions
  - 3.9.2. Filtering data in repositories
  - 3.9.3. Discussion on efficiency
- 3.10. Saga/Process manager – business process model
  - 3.10.1. Saga management framework design
  - 3.10.2. Technical traps
- 3.11. Role Object
  - 3.11.1. Role model – when the system should behave (not just look) differently depending on the domain role of the user
- 3.12. Additional patterns - Building Blocks extensions
  - 3.12.1. Policy Decorators - Supple Design
  - 3.12.2. Aggregate as a state machine
  - 3.12.3. Chain of responsibility
  - 3.12.4. Builder

4. Command-query Responsibility Segregation Architecture

- 4.1. CqRS as a development of a layer architecture
  - 4.1.1. Concept of intentional GUI
  - 4.1.2. Two stacks of layers
- 4.2. Write Stack

BO·TT·EGA
IT minds

BOTTEGA IT minds