DOMAIN-DRIVEN DESIGN
FOR MICROSERVICES

FUNDAMENTAL

OLIVIER GIERKE
OGIERKE@PIVOTAL.IO

Pivotal
Oliver Gierke
olivergierke

Spring Data Project Lead @ Pivotal, OpenSource enthusiast, all things Spring, data, DDD, REST and software architecture. Musician, producer... Soul Power!

- Pivotal Software, Inc.
- Dresden, Germany
- info@olivergierke.de
- http://www.olivergierke.de
- Joined on 18 Sep 2009

797 Followers 56 Starred 31 Following

Overview
Repositories
Public activity

Pinned repositories

- spring-projects/spring-data-examples
  Spring Data Example Projects
  602 ★

- spring-projects/spring-data-jpa
  Simplifies the development of creating a JPA-based data access layer.
  661 ★

- spring-bucks
  Implementation of the sample from REST In Practice based on Spring projects
  357 ★

- spring-projects/spring-data-rest
  Simplifies building hypermedia-driven REST web services on top of Spring Data repositories
  393 ★

- rest-microservices
  Sample for Spring Boot based REST microservices
  87 ★

1,493 contributions in the last year

Summary of pull requests, issues opened, and commits. Learn how we count contributions.

Contribution settings →
http://www.infoq.com/minibooks/domain-driven-design-quickly

A Summary of Eric Evans' *Domain-Driven Design*

**Domain-Driven Design** *Quickly*
Microservices ♥
Domain-Driven Design

11:10 — Beethoven Saal
Implementing Domain-Driven Design
Value objects
public class Customer {

    private Long id;
    private String firstname, lastname, email;

    ...
}

Stringly typed code
public class SomeService {

    public void createUser(String firstname,
                               String lastname, String email) {

        ...
    }

}
public class Customer {

    private Long id;
    private Firstname firstname;
    private Lastname lastname;
    private EmailAddress emailAddress;

    ...

}
Value Objects are a PITA to build in some languages.
Still, they’re worth it.

See “Power Use of Value Objects in DDD” by Dan Bergh Johnsson.
Lombok — putting the spice back into Java.
@Value
public class Customer {

    UUID id = UUID.randomUUID();
    Firstname firstname;
    Lastname lastname;
    EmailAddress email;

    @Value
    static class EmailAddress {
        String value;
    }
}
AutoValue

Project website on GitHub.
Entities & Aggregates
Persistence technology VS. Domain model
Aggregate = Entity + Repository
Bounded Context
Domain-Driven Design & Monoliths
Avoid technologies that get in your way.
How to enforce context boundaries?
What about consistency?
References between Bounded Contexts?
Domain Events
Level 0: No events at all
Level 0: No events at all

Level 1: Explicit operations
If you’re calling two setters in a row, you’re missing a concept.
Level 0: No events at all

Level 1: Explicit operations

Level 2: Some operations as events
State transitions become domain events.
Level 0: No events at all

Level 1: Explicit operations

Level 2: Some operations as events

Level 3: Event Sourcing
Domain-Driven Design & Microservices
Bounded contexts define system boundaries.
Architecture is less likely to deteriorate as it’s harder to violate boundaries.
Restructuring service boundaries is much harder.
Inter-Service communication becomes remote communication.
Accept and embrace eventual consistency between services.
Domain Events
REST & Messaging
Domain events become state transitions.
Hypermedia for state transition and explicit events.
Thanks!