BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

A DDD example with Java, Spring, Spring Data JPA, Derby

Trying to understand
Domain Driven Design

Christoph Knabe
Dpt. Informatics and Media
20.07.2017

Inhalt

- Bio
- Why is DDD interesting?
- Study Sources
- Good about DDD
- What layering
- Layering and Entity Classes
- Immutable Value Objects and JPA
- Conclusion
- Demonstration

Biography

- 1981-1990 Software Developer @ PSI GmbH:
  Factory Automation, Software Engineering Tools

- 1990-… Prof. @ Beuth University of Applied Sciences Berlin:
  Teaches Software Engineering, Programming
  Main interests: Scala, Backend Development

Why is DDD interesting for me?

- First contact about 2006. Read the book of Eric Evans, did not catch me.
- Second contact in 2016. Mentioned in discussions of Scala usergroups. Still modern!
  Had a closer and newer look.
- Have to teach Software Engineering. Hopefully not only UML.

What did I study now?

- **Naked Objects** approach with tool Apache ISIS:
  Pro: Model domain as Java classes,
  generate REST service and UI.
  Contra: Needs Lombok plugin, maybe too complicated
  for students, not mainstream.

- Official DDD example projects:
  - Spring Cargo Tracker
  - Java EE Cargo Tracker
  - some web articles

What is good about DDD?

- DDD avoids „Anemic Domain Model“ (stupid public getters and setters)
- Real object-oriented Data Abstraction
- An **attractive name** for this!
  (All students, to whom I recommended the Rich Domain Model of Martin Fowler ended up with anemic domain classes, as there are many more recommendations for this in the web.)

What layering?

- **Classical**: 3 layers:
  UI → Logic → Data Access
  Inevitable with static binding.
- **Eric Evans DDD (2003)**: 4 layers:
  Interface → Application → Domain → Infrastructure
  That is why I did not understand his book!
- **Modern** (Spring Cargo Tracker): 4 layers:
  Infrastructure → Interface → Application → Domain

How to do it?

- **How can the domain model be the lowest layer?**
  It must use the Infrastructure!

- **Solution (Ports and Adapters pattern)**:
  Domain model offers domain services to the upper layers.
  Domain model requires interfaces for persistence etc.
  They are injected by a container.

How to do it in entity classes?

- **Rich Entity class must have access to persistence services!!!**
  DI can't inject it when retrieving entities from JPA, ...

- **Manual after-retrieve enrichment?**
  After retrieving a collection of entities from persistence fill the persistence services references into all entities.
  Domain model requires interfaces for persistence etc.

- **DODI:** Spring Domain Object Dependency Injection?
  Needs AOP Load Time Weaving.
  Maybe too complicated for students.

Would prefer immutable Value Objects

- **But JPA can't restore them!!!**
  Should I leave them mutable???
- **Or should I use Hibernate?**
  Seems to support immutable embedded objects.

Conclusion

- **DDD seems good for my purpose**:
  Real abstraction
- **Problem, how to make it simple.**
- **Now going to demonstrate on code in the IDE.**

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Thank You