
ORACLE®

ORACLE

Domain Driven Design with Java EE 6

Reza Rahman
Java EE/GlassFish Evangelist

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

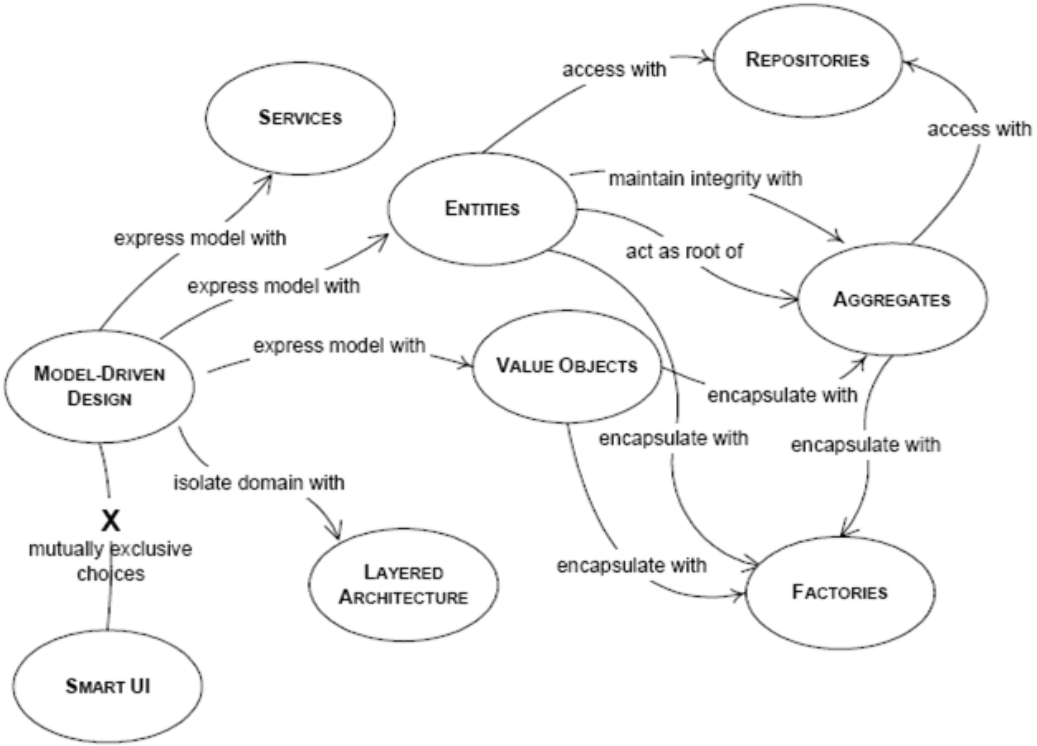
Program Agenda

- Domain Driven Design Overview
- Java EE 6/GlassFish Domain Driven Design Demo

Domain Driven Design/Java EE

- J2EE popularized family of layered server-side application architectures - J2EE Blue Prints (aka “Java Pet Store”)
- J2EE Blue Prints encumbered with implementation details due to immaturity of technology, particularly EJB 2.x
- With rise of POJO programming, domain driven design (DDD) emphasizes a return to Object Oriented Analysis and Design (OOAD)
- Java EE’s lightweight programming model fits the DDD philosophy well

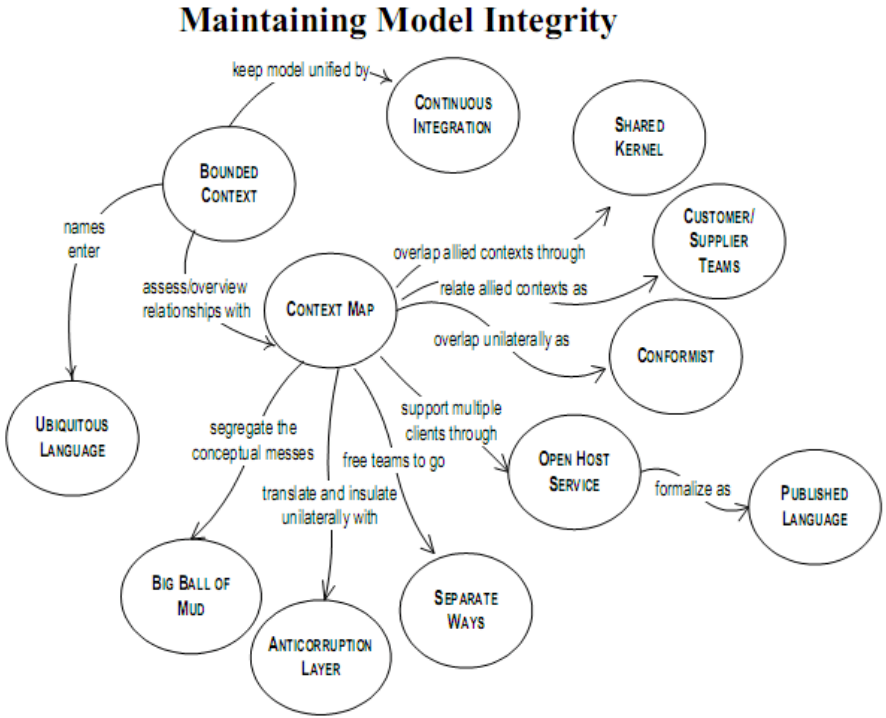
Domain Driven Design



Core Concepts

- **Domain:** A sphere of knowledge, influence, or activity. The subject area to which the user applies a program is the domain of the software
- **Model:** A system of abstractions that describes selected aspects of a domain and can be used to solve problems related to that domain
- **Ubiquitous Language:** A language structured around the domain model and used by all team members to connect all the activities of the team with the software
- **Context:** The setting in which a word or statement appears that determines its meaning

Strategic Domain-Driven Design



Building Blocks

- **Entity:** An object that is not defined by its attributes, but rather by a thread of continuity and its identity
- **Value Object:** An object that contains attributes but has no conceptual identity. They should be treated as immutable
- **Aggregate:** A collection of objects that are bound together by a root entity, otherwise known as an aggregate root. The aggregate root guarantees the consistency of changes being made within the aggregate by forbidding external objects from holding references to its members

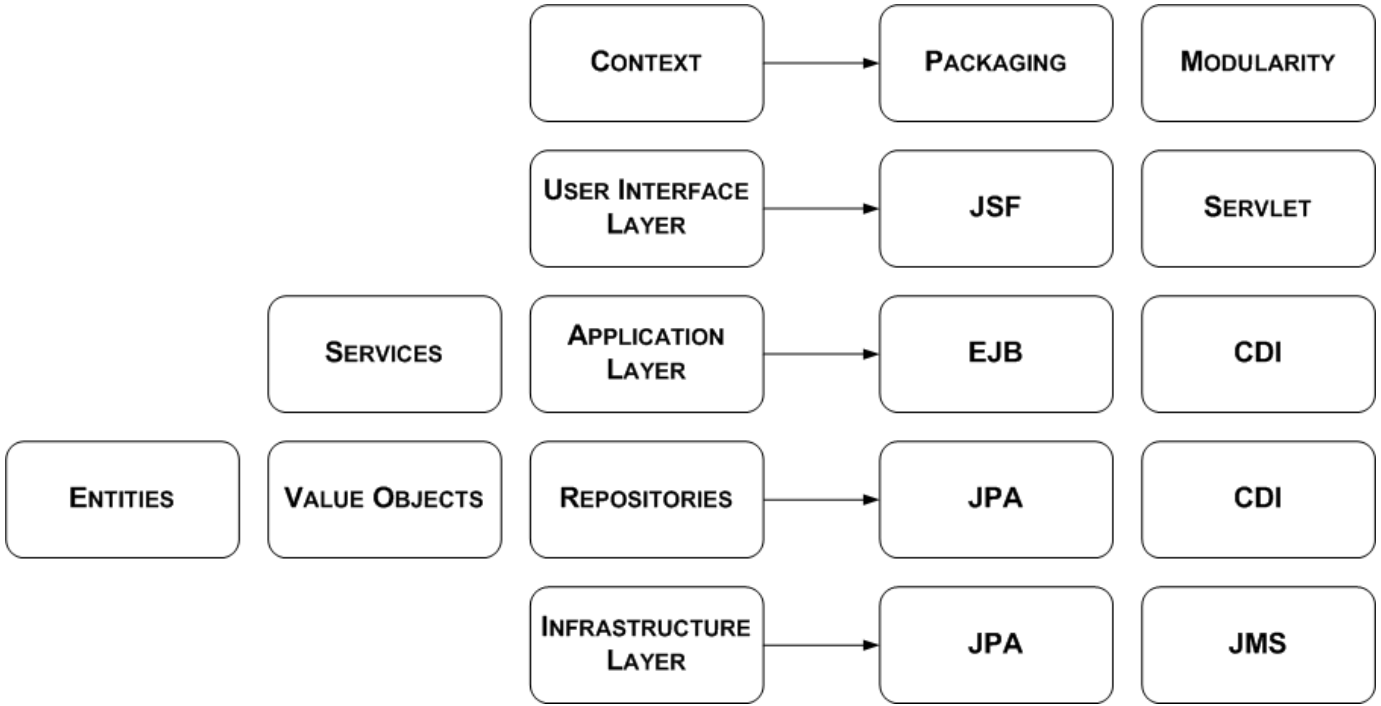
Building Blocks (Continued)

- **Service:** When an operation does not conceptually belong to any object. Following the natural contours of the problem, you can implement these operations in services
- **Repository:** Methods for retrieving domain objects should delegate to a specialized Repository object such that alternative storage implementations may be easily interchanged
- **Factory:** Methods for creating domain objects should delegate to a specialized Factory object such that alternative implementations may be easily interchanged

Layers

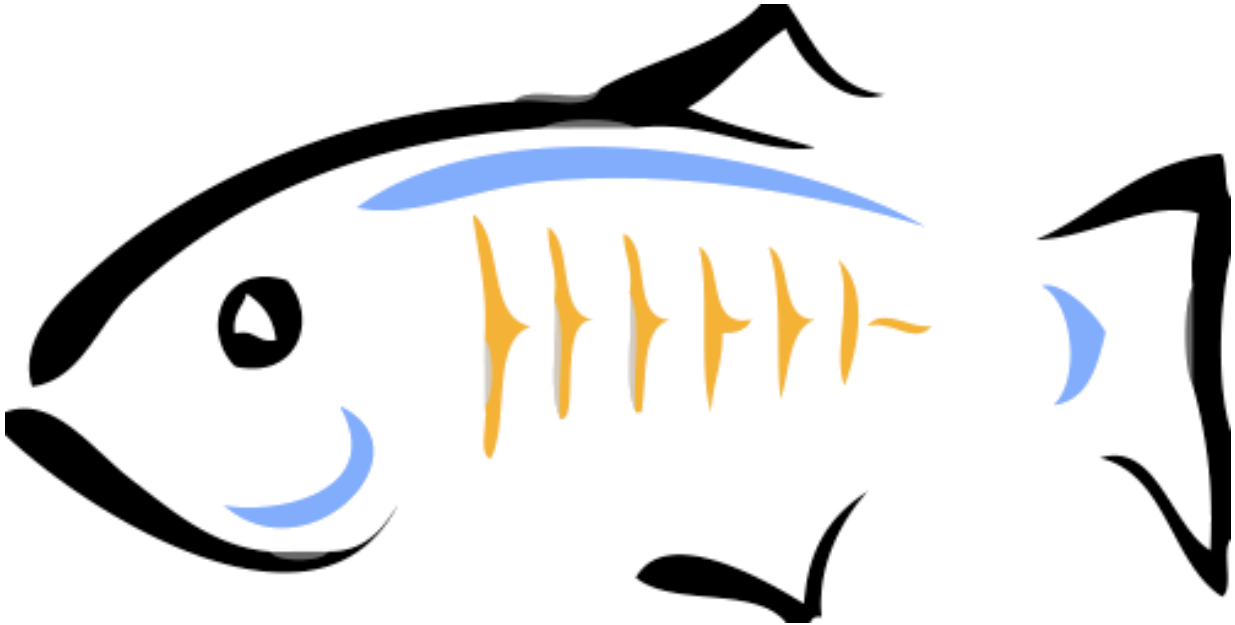
- **User Interface:** Shows the user information and receives input
- **Application:** Thin layer to coordinate application activity. No business logic or business object state is in this layer
- **Domain:** All the business objects and their state reside here. Objects in the domain layer should be free of concern about displaying or persisting themselves
- **Infrastructure:** The objects dealing with housekeeping tasks like persistence.

Mapping DDD to Java EE



Java EE 6 DDD Demo

Try it Out!



glassfish.org/downloads

Resources

- Getting Started with Domain-Driven Design
  - http://refcardz.dzone.com/refcardz/getting-started-domain-driven
- Domain Driven Design Quickly
  - http://www.infoq.com/minibooks/domain-driven-design-quickly
- DDD Sample
  - http://dddsample.sourceforge.net
- Java EE 6 Tutorial
  - http://docs.oracle.com/javaee/6/tutorial/doc/
- The Aquarium
  - blogs.oracle.com/theaquarium

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®