Creating Lightweight Applications

...with Nothing but Vanilla Java EE 6

adam-bien.com / twitter: @AdamBien

- Expert Group Member (jcp.org) of Java EE 6, EJB 3.1, Time and Date and JPA 2.0

- Java Champion, (JavaONE) speaker + rockstar, freelancer, consultant and author: >100 articles, 7 German books

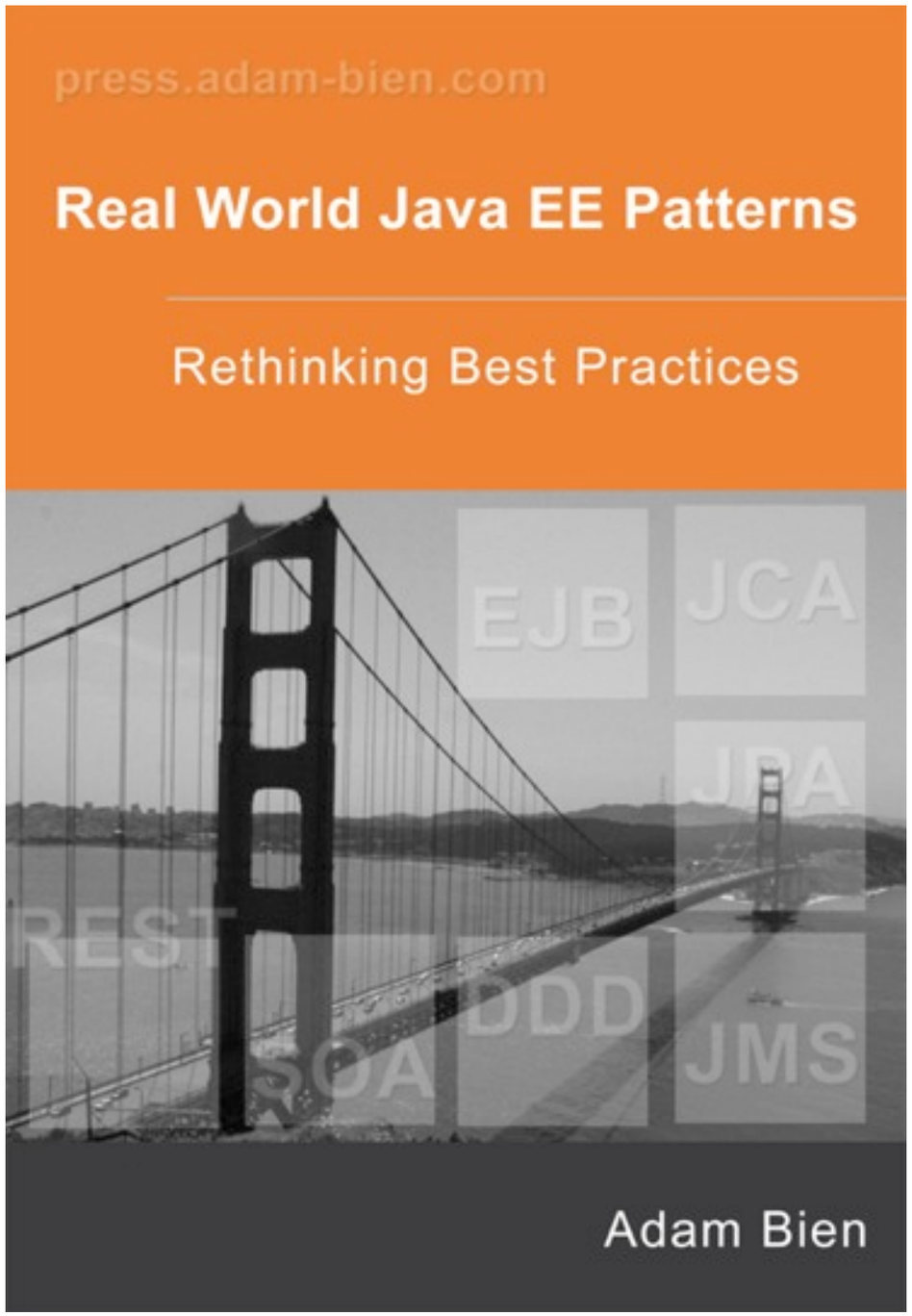
- “Real World Java EE Patterns– Rethinking Best Practices” http://press.adam-bien.com

- Consultant, Developer and Architect (since JDK 1.0)

- Project owner/committer:

  http://kenai.com/projects/javaee-patterns/

adam-bien.com



press.adam-bien.com

adam-bien.com

...start with a humble statement

adam-bien.com

Java EE 6 - The simplest possible and most lightweight platform you can currently get

adam-bien.com

Demo

- The feel of Java EE 6:
  - Incremental Deployment
  - Convention over Configuration
  - Dependency Injection
  - REST, EJB 3.1, Events, JPA, Validation...

Conclusion

- Java EE 6 application mainly consists of ...business logic.
- Resulting design is leaner than POJOs
- Any simplification is hardly possible
- Most of the J2EE Patterns and Best Practices are ...deprecated.

Demo Conclusion

- WAR is the new EAR
- Deployment units (EAR / WARs) are < 1 MB
- Milliseconds (re)deployment is usual
- Java EE 6 apps are application server independent

What is Lightweight?

adam-bien.com

Lightweight

- Fast
- Simple
- Small
- Lean
- Short turnaround cycles
- “Easy”

Domain Driven Strategy

adam-bien.com

DDD

- Business (Domain) first
- Encapsulated objects with behavior
- Rich domain model
- Almost no logic left for services
- Stateful architecture
- Fine grained services and interactions

DDD

- Emphasizes entities, not services
- Fits well with ROA (REST) and the notion of resources
- Lean: consists only of few layers - state and behavior are collapsed.

Service Driven Strategy

adam-bien.com

Service Driven Strategy

- Anemic domain objects (getters / setters only)
- Rich service layer
- Procedural programming style
- Stateless architecture is beneficial
- Coarse grained interfaces

Service Driven Strategy

- Emphasizes services, not objects
- Works well with SOAP and the notion of SOA
- Consists of several layers with separated state and behavior

SOA <=> DDD

adam-bien.com

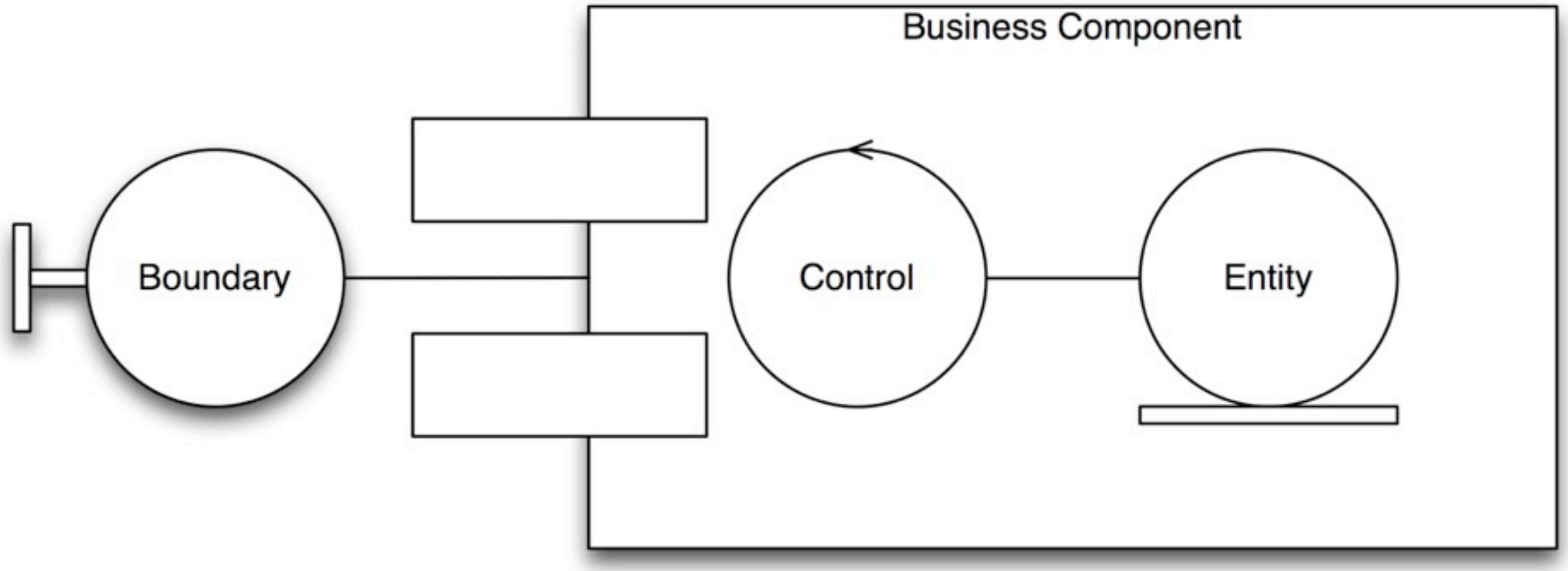
SOA <=> DDD

- Both strategies are best practices for an appropriate context
- ...DDD is exactly the opposite of SOA
- In an average project both strategies can be applied

Simplest Possible Component

adam-bien.com

ECB



adam-bien.com

CDI (JSR-299)

- CDI is lacking some EJB 3.1 features:
  - Pooling: important for throttling and prevention of denial of service attacks
  - Monitoring - all EJBs are exposed via JMX and can be so monitored with JConsole
  - Timers
  - Asynchronous, transactional execution with Future support
  - Declarative, "Convention Over Configuration" transactions.

CDI (JSR-299)

- EJB 3.1 as a boundary is the "KISSiest" possible choice
- CDI comes with powerful, typesafe Dependency Injection, Events (Publish-Subscribe), aspects and glue logic
- EJB 3.1 + CDI are the best possible combination in "vanilla" Java EE 6

Java EE 6 Mapping

- **Boundary**: (Stateless) (no-interface view) Session Bean
- **Control** [optional]: managed CDI-bean
- **Entity**: JPA 2 Entity

Boundary

adam-bien.com

Boundary

- Well-defined entry point
- Separation between UI and business logic

- A public API of a module, component, package
- Implements business logic OR coordinates controls.

Boundary

- Decorated with (EJB 3.1) aspects:
  - transactions,
  - remoting (REST, Hessian, IIOP),
  - monitoring,
  - concurrency,
  - asynchronous processing, custom aspects

Control

adam-bien.com

[Control] - optional

- A CDI managed bean with dependent scope, or no-interface view EJB 3.1
- Reusable services or business logic
- Is hidden behind a boundary
- Implements Entity-independent business logic and cross-cutting functionality

Control contd.

- Replaces the DAO pattern
- Groups reusable queries and data access logic
- Wraps and decorates EntityManager
- Is **optional**: CRUD and simple data operations can be implemented directly in the Boundary

Entity

adam-bien.com

Entity

- “Just” a persistent object.
- Unifies state and behavior.
- Also executable (testable) without Control / Boundary.
- A Rich Domain Object.

CDI - EJB -CDI (CEC)

adam-bien.com

CEC

- CDI is perfect for backing bean implementation (value binding, @Named etc.)
- EJB 3.1 do provide obligatory cross-cutting concerns without any noise (=perfect facades)
- CDI is nice for the implementation of fine grained business logic again

Conclusion 2

adam-bien.com

Conclusion 2

- Most of the J2EE Patterns are obsolete
- The realization of GoF Patterns with Java EE 6 is leaner, than with Java SE
- Java EE 6 is a lean and productive standard - not only suitable for the “enterprise”

Pragmatism + Java EE 6 = Hacking On Steroids

adam-bien.com

Thank You!

blog.adam-bien.com
twitter.com/AdamBien