



# Object persistence with Hibernate in Decision Deck 1.1

Gilles Dodinet

2<sup>nd</sup> Decision Deck Workshop  
2008, 21-22 February



## D2 1.1 changes overview

- Move from manual SQL writing to a persistence managed solution
  - JPA, Hibernate
  - Simplicity & Productivity
  - Robustness (wrt model changes)
- Standard-based plugin mechanism
  - Based on OSGi: highly dynamic component specification
- Unified model event mechanism
  - « Let me know when *it* is changed »



# Agenda

- D<sup>2</sup> 1.0.x architecture recap
  - Persistence handling
  - Modularity concerns
  - Plugin descriptor
- D<sup>2</sup> 1.1 Changes
  - ORM and JPA overview
  - Persistence handling with Hibernate
  - Getting your feet wet
  - Induced changes
- In practice
  - Decision Deck Eclipse plugin



## D<sup>2</sup> 1.0 : architecture recap

- Object persistence: Raw JDBC
- Modularity concerns: Classworld
- Plugin description: plugin.xml



## Persistence handling in D<sup>2</sup> 1.0.x

- Raw JDBC
  - Verbose
  - Broke the DRY principle
  - Tedious, repetitive task
  - But: as powerful as needed
- Custom SQL Framework
  - No multiple joins
  - Missed a few SQL constructs
  - Almost as verbose as Raw JDBC



## Modularity concern in D<sup>2</sup> 1.0.x

- Plugin isolation is a requirement
- Classloading issues dealt with Classworld
- Drawbacks
  - Custom management of native libraries
  - No « Buddy-like » mechanism
- Main OSGi features reimplemented



# Plugin descriptor in D<sup>2</sup> 1.0.x

```
<plugin>
  <id>org.decisiondeck.MyPlugin</id>
  <version>1.0</version>
  <className>org.decisiondeck.my.MyPlugin</className>
  <services> ..... </services>
  <requires>
    <require>
      <id> ... </id>
      <packageNames> ... </packageNames>
    </require>
  </requires>
  <refines>
    <tableAdditions> <!-- VERY VERBOSE --> </tableAdditions>
  </refines>
</plugin>
```



## D<sup>2</sup> 1.1 : What changed ?

- Persistence now handled with Hibernate
- Border-side consequences:
  - We moved from Classworld to OSGi
  - Plugin descriptor is now much lighter
  - Service layer does not act anymore as a global object cache
- Minor changes
  - All projects stored in the same database
  - A few API changes (naming)



# Object Relational Mapping

- The Impedance Mismatch
  - How to reconcile two (concurrent) different (logical) representations of the same concept?
- Solving the Impedance Mismatch
  - Either manually through raw JDBC (and raw SQL)
  - Or with the help of *ORM frameworks*
- In either case there are a few common patterns to be aware of:
  - How to map inheritance : *single table, one table per concrete class, join tables*
  - How to map relationships : *one-to-many, many-to-many, many-to-one*



# Java Persistence Api

- Initial objective
  - Simplify EJB CMP entity beans development
  - Developed as part of JSR-220
- JPA usable in J2EE as well as in J2SE
- In a nutshell
  - ORM layer standardization
  - Mapping description
    - Java 5 annotations and/or XML
  - Data Access API
    - Pluggable providers (Hibernate, Toplink, JPox, etc.)
  - SQL-Like query language



# Object persistence in D2 1.1

- Database query through Hibernate Core API
  - Criteria API available
  - Control session (un)loading
- Mapping specified with JPA annotations
- Persistent entities registered in plugin.xml
- Generic Accessor provided
  - Simple CRUD
  - A few aggregate functions (count, ..)



# Hibernate Session object

- Quoting hibernate documentation:
  - The *Session* « is the central API class abstracting the notion of a persistence service. »
- Manages entities lifecycle
  - Transient à Persistent à Detached
- Main operations:
  - save(), update(), delete(), merge()



## Using Hibernate in D<sup>2</sup> 1.1

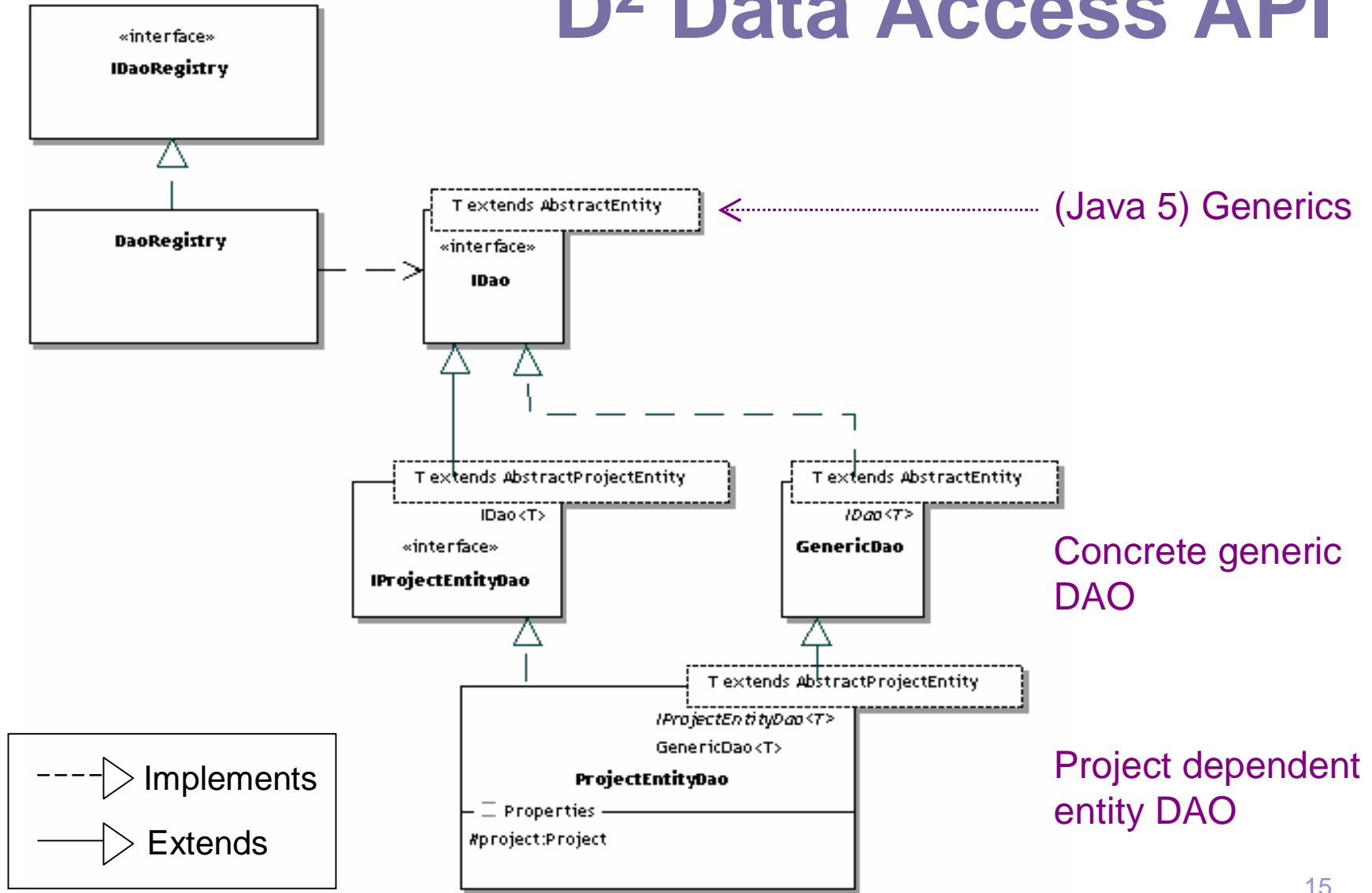
- A single Hibernate Session is used throughout the whole application
- Session is opened when opening a project, closed when closing a project
- Most of Hibernate complexity is hidden in GenericDao and DaoRegistry
- The current Session can be easily retrieved via the GenericDao *getSession()* method



# Querying with Hibernate: HQL

- **Hibernate Query Language**
  - Query language similar to SQL
  - Object oriented
- **Criteria API**
  - A simple object oriented way to build queries
  - A simple object oriented way to build queries
- **In either case**
  - Support for aggregate functions

# D<sup>2</sup> Data Access API





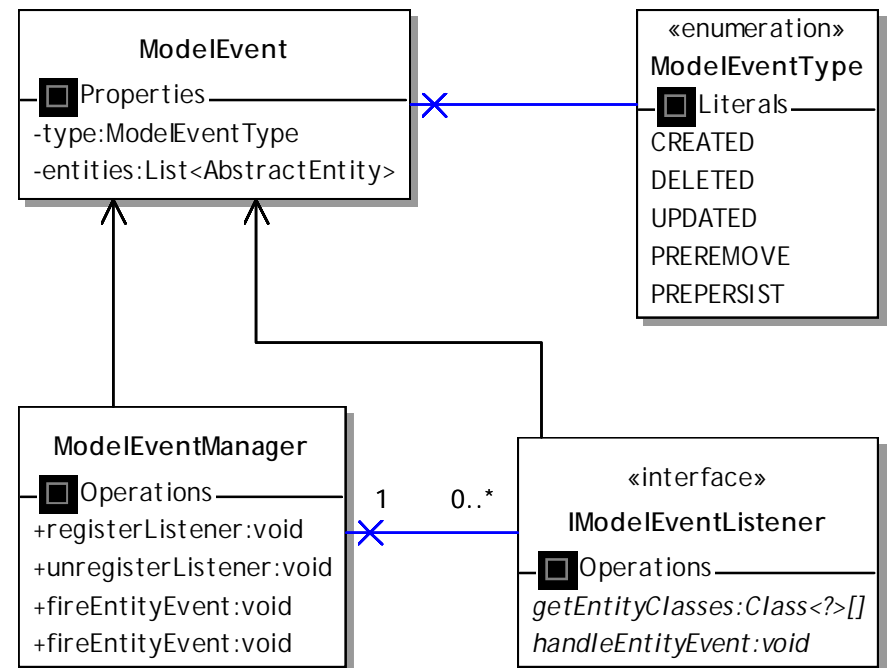
# Dao Registration

- Plugin's DAO must be registered
  - The registry can instantiate them
  - They can be retrieved by key
  - DaoRegistry#register(key, classname)
- They must be unregistered to clean up resources
  - DaoRegistry#unregister(key)
- Plugin class controls the plugin lifecycle
  - Registration occurs during plugin startup
  - Deregistration occurs at plugin shutdown



# Model Events

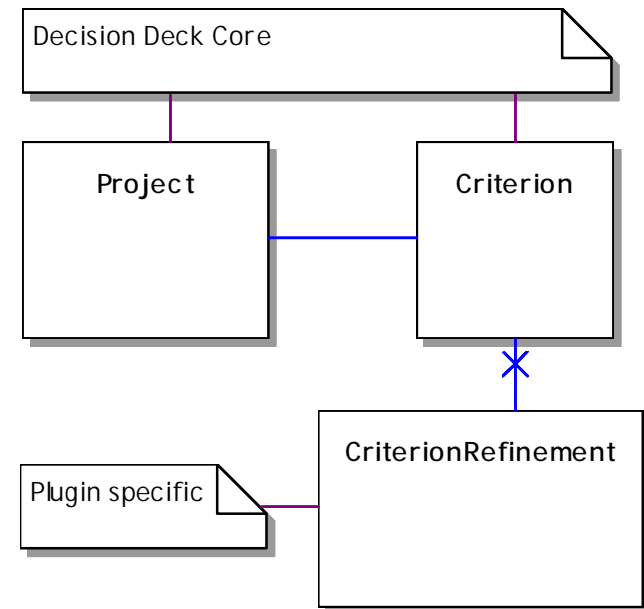
- PropertyChanged event not used anymore
- Centralized « Model Events » management
  - Homogenization
  - Strongly typed
  - Easier debugging





# Handling references

- Fact
  - Hibernate imposes strong relational integrity (good practice)
- Example
  - *Criterion* does not know anything about *CriterionRefinement*
  - *Criterion* deletion may fail if there exist some refinement
- Resolution
  - *CriterionRefinement* provider must register a *ModelEvent* listener to monitor *Criterion* deletions
  - Throw *VetoException* if needed





# Induced changes

- Problem
  - Hibernate needs to see the persistent classes
  - D<sup>2</sup> 1.0.x classloading relies on Classworld which does not allow two-way visibility by default
- Solutions
  - Severely refine classloading mechanism
  - Drop Classworld in favor to OSGi
- OSGi
  - « The Dynamic Module System for Java™ »
  - Covers all our classloading needs
  - Equinox Buddy mechanism allows to bypass classloader isolation default rules
  - Unit of deployment is called a *bundle*



# OSGi in D<sup>2</sup> 1.1

- Advantages
  - Greatly simplify classloading management
  - Service-oriented platform
- Bundle configuration
  - MANIFEST.MF
    - Especially defines the dependencies: in D<sup>2</sup> 1.0.x they were defined through the <required> tag in plugin.xml
  - plugin.xml : D<sup>2</sup> specific configuration
    - Identification
    - Services
    - Entities
- Eclipse D<sup>2</sup> Plugin wizard
  - Generates a full functional example



# Questions ...



# Resources

- JPA on the desktop
  - <http://java.sun.com/developer/technicalArticles/J2SE/Desktop/persistenceapi/>
- Getting started with Hibernate
  - <http://www.hibernate.org/152.html>
- OSGi in a nutshell
  - <http://gravity.sourceforge.net/servicebinder/osginutshell.html>
- D<sup>2</sup> plugin migration guide
  - <http://decision-deck.sourceforge.net/migrate/migration-guide.html>