Easy EJB

An Introduction to the Simplified World of Enterprise JavaBeans 3.0

Scott Crawford
www.scottcrawford.com

Delegate Pre-Requisites

To benefit from this tutorial:

- You should be comfortable with object-oriented programming concepts (class, object, inheritance, etc.) and able to read simple Java examples
- You should be comfortable with the idea of a relational database and its use of tables and SQL

You do NOT require any experience of Enterprise JavaBeans (EJB) technology.

A Flying Start

A Java Class

```
public class HelloEasyWorldBean {

    public String meetPerson(String firstName, String lastName) {
            return "Hello " + firstName;
    }

    public String meetCouple(String firstName, String anotherFirstName,
                                                        String lastName) {
            return "Hello " + firstName + " and " + anotherFirstName;
    }

}
```

Another Java Class

```
public class Contact {

    private Long id;
    private String firstName;
    private String lastName;

    public Contact() {
    }

    public Contact(String firstName, String lastName) {
        setFirstName(firstName);
        setLastName(lastName);
    }
```

[continues]…

…[continued]

```
    public Long getId() {
        return id;
    }

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }


    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}
```

HelloWorld Creates Contacts

```
public class HelloEasyWorldBean {

    public String meetPerson(String firstName, String lastName) {
        Contact newFriend = new Contact(firstName, lastName);
        return "Hello " + firstName;
    }

    public String meetCouple(String firstName, String anotherFirstName,
                                            String lastName) {
        Contact oneFriend = new Contact(firstName, lastName);
        Contact anotherFriend = new Contact(anotherFirstName,
                                                        lastName);
        return "Hello " + firstName + " and " + anotherFirstName;
    }
}
```

What should we add next to make this little program more useful?

Contact Class Becomes an EJB

```
@Entity
public class Contact {

    [...as before...]

    @Id(generate=GeneratorType.AUTO)
    public Long getId() {
        return id;
    }

    [remainder as before...]
}
```

HelloWorld Class Becomes an EJB

```
@Stateless
public class HelloEasyWorldBean {

    @Inject private EntityManager em;

    public String meetPerson(String firstName, String lastName) {
            Contact newFriend = new Contact(firstName, lastName);
            em.persist(newFriend);
            return "Hello " + firstName;
    }
```

[continues…]

HelloWorld Class Becomes an EJB

[…continued]

```
    public String meetCouple(String firstName, String anotherFirstName,
                                            String lastName) {
        Contact oneFriend = new Contact(firstName, lastName);
        Contact anotherFriend = new Contact(anotherFirstName,
                                                   lastName);
        em.persist(oneFriend);
        em.persist(anotherFriend);
        return "Hello " + firstName + " and " + anotherFirstName;
    }
}
```

Tutorial Structure

- A Flying Start (We did that already)
- Introduction
- Themes of EJB 3.0
- Demo of a Working System
- More themes from the Demo

Introduction

The Vision in 1999…

The Enterprise JavaBeans architecture is a component architecture for the development and deployment of component-based distributed business applications. Applications written using the Enterprise JavaBeans architecture are scalable, transactional, and multi-user secure. These applications may be written once, and then deployed on any server platform that supports the Enterprise JavaBeans specification.

from the Enterprise JavaBeans spec, version 1.1 (final)

…and as Revised in 2005

An entity bean is a lightweight persistent domain object.

from the Enterprise JavaBeans spec, version 3.0 E.D. 2

J2EE and EJB Version Numbers

J2EE version	1.2	1.3	1.4	5.0
When	Dec 1999	July 2001	Nov 2003	Q1 2006
EJB version	1.1	2.0	2.1	3.0
BEA WebLogic version	6.1	7.0, 8.1	9.0	
IBM WebSphere version	4.0	5.0	6.0	

Terms of Reference

The purpose of EJB 3.0 is to reduce complexity from the EJB developer’s point of view.

EJB 3.0 is just beyond the Early Draft stage. It is being developed officially as JSR-220 under the Java Community Process.

Therefore it is possible that anything in this presentation may change prior to the final specification.

EJB 3.0 Timetable

Spec group formed	Sep 2003
Early Draft 1	June 2004
Early Draft 2	February 2005
Public Draft	*expected June 2005*
Final Draft	*expected Q1 2006*

Context for Enterprise JavaBeans 3.0

EJB3 will be part of J2EE version 5

The new POJO Persistence Model can be:

- Deployed in a J2EE application server, OR
- Used independently with a persistence implementation

EJB3 is built upon Java core language version 5 (“Tiger”), allowing use of such features as:

- Metadata/Annotations
- Generics
- Enhanced for loops

Themes of EJB 3.0

Some Themes of EJB 3.0

- *Annotations*
- *A non-intrusive, POJO programming model*
- *Defaulting and configuration-by-exception*
- *Facilitation of test-driven development*
- *Lightweight POJO persistence model*
- *Object/Relational mapping*
- *A portable *and* powerful query language*

Java 5 Annotations

- @Stateless
- @Stateful
- @MessageDriven
- @Entity

- @Remote, @Local
- @Id, @Inject

POJO Programming Model - 1

- Bean implementations are Plain Old Java Objects
- They are not required to implement any mandated, burdensome interfaces
- Checked exceptions are avoided

POJO Programming Model - 2

- Home interfaces are eliminated entirely
- Although Sessions and MDB’s still require a business interface…
  - they use the normal [class] implements [interface] Java mechanism
  - and may be auto-generated in simple cases

Configuration by Exception

- Session and MDB business methods default to container-managed, required transaction settings – you only specify when you want otherwise
- Local interfaces are assumed, and, if you wish, generated for you
- Standard but powerful object-relational mapping strategies are defaulted but override-able
- Many systems will not require deployment descriptors

Test-Driven Development

```
@Stateless
public class HelloEasyWorldBean {
…

    @Inject private EntityManager em;
```

OR

```
    private EntityManager em;

    @Inject
    public void setEntityManager(EntityManager em) {
            this.em = em;
    }
```

Lightweight POJO Persistence - 1

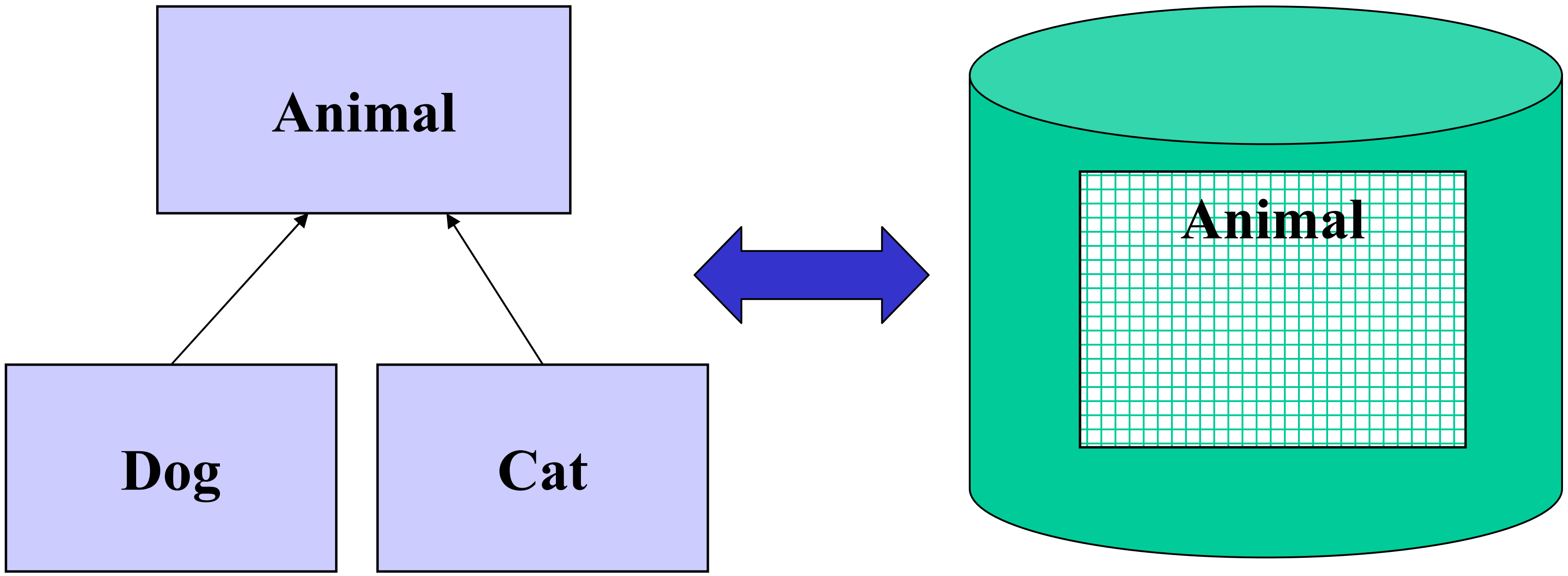
```
@Inject private EntityManager em;

public String meetPerson(String firstName, String lastName) {
        Contact newFriend = new Contact(firstName, lastName);
        em.persist(newFriend);
        return "Hello " + firstName;
}
```

- Natural, universal creation pattern
- Inheritance and Polymorphism fully supported

Lightweight POJO Persistence - 2

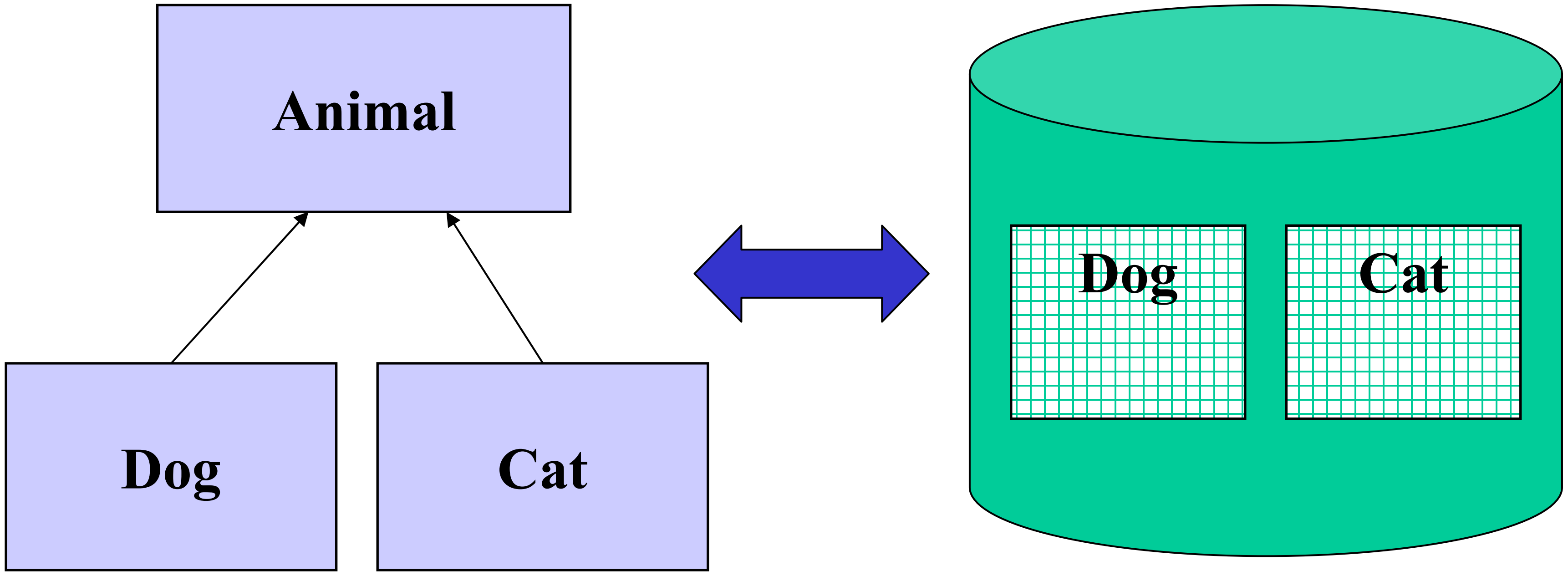
```
package javax.persistence;

public interface EntityManager {

    public void persist(Object entity);
    public void remove(Object entity);

    public <T> T find(Class<T> entityClass,
                      Object primaryKey);

    public Query createQuery(String ejbqlString);

    [and several other methods]
```

Object/Relational Mapping - 1



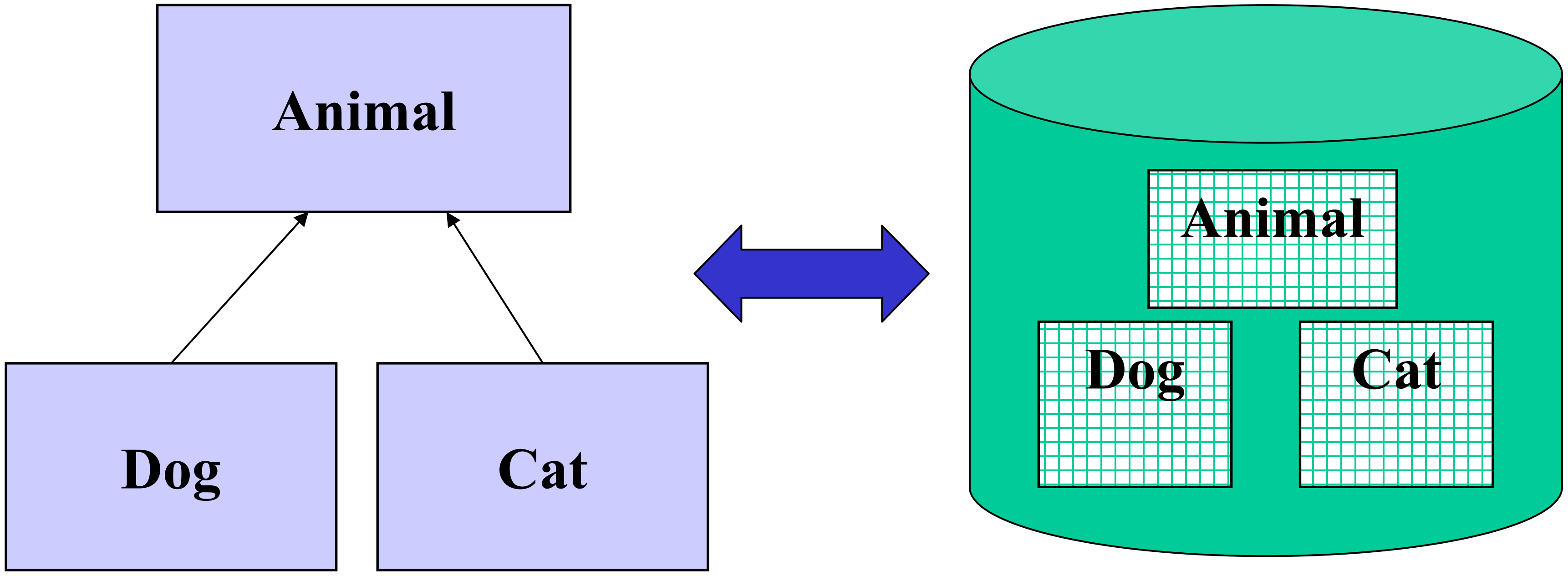
@Inheritance(strategy=SINGLE_TABLE)

Object/Relational Mapping - 2



@Inheritance(strategy=TABLE_PER_CLASS)

Object/Relational Mapping - 3



@Inheritance(strategy=JOINED)

Powerful Query Language

- Static or dynamically-defined queries
- Queries are polymorphic across inheritance hierarchies
- Bulk update and delete operations
- Pagination capability is standard
- Control over eager/lazy loading
- Many SQL features and syntax imports

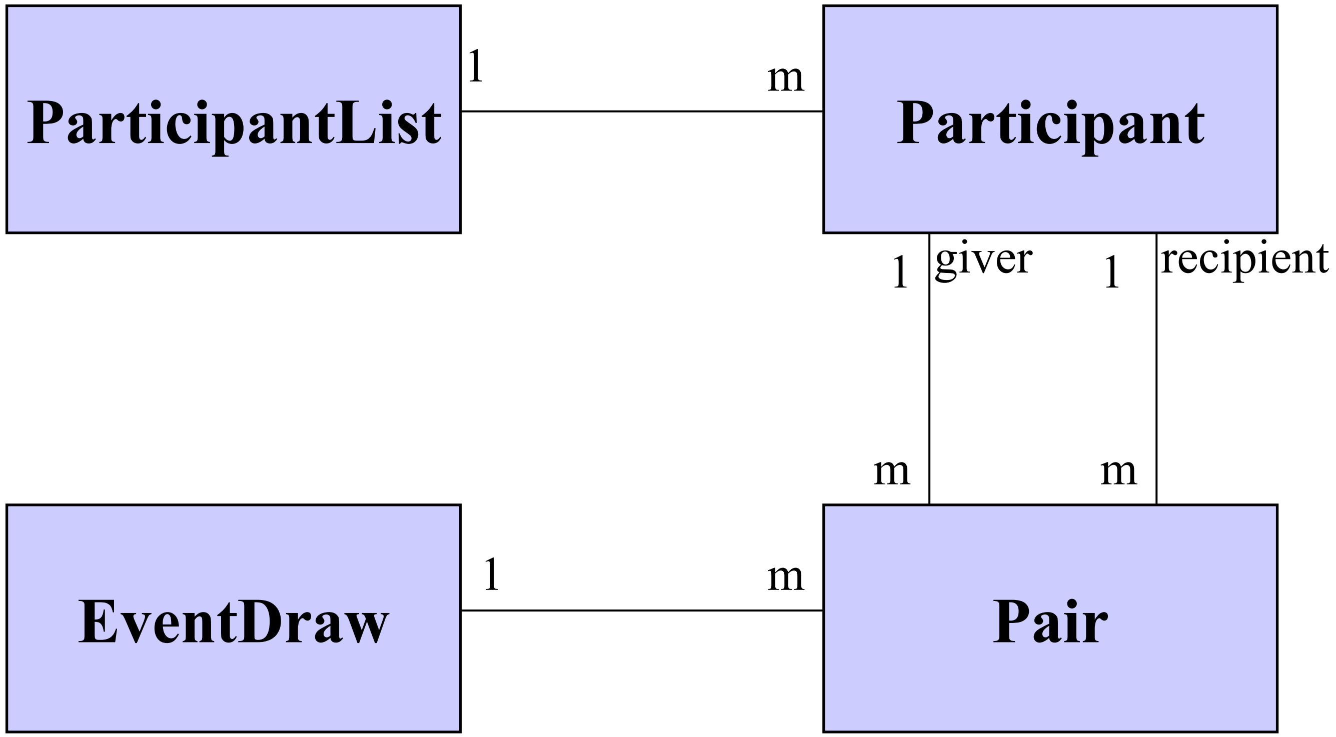
Demo

More Themes from the Demo

More Themes from the Demo

- The Abstract Schema
- Entity Relationships
- Session Facades are Still in Style
- Entity Beans Travel
- Entity Lifecycle
- Local and Remote Interfaces
- Small and Fun Codebase

Abstract Entity Schema



Bidirectional ManyToOne Relationship

In Participant.java:

```
@ManyToOne
public ParticipantList getParticipantList()
{
      …
```

In ParticipantList.java:

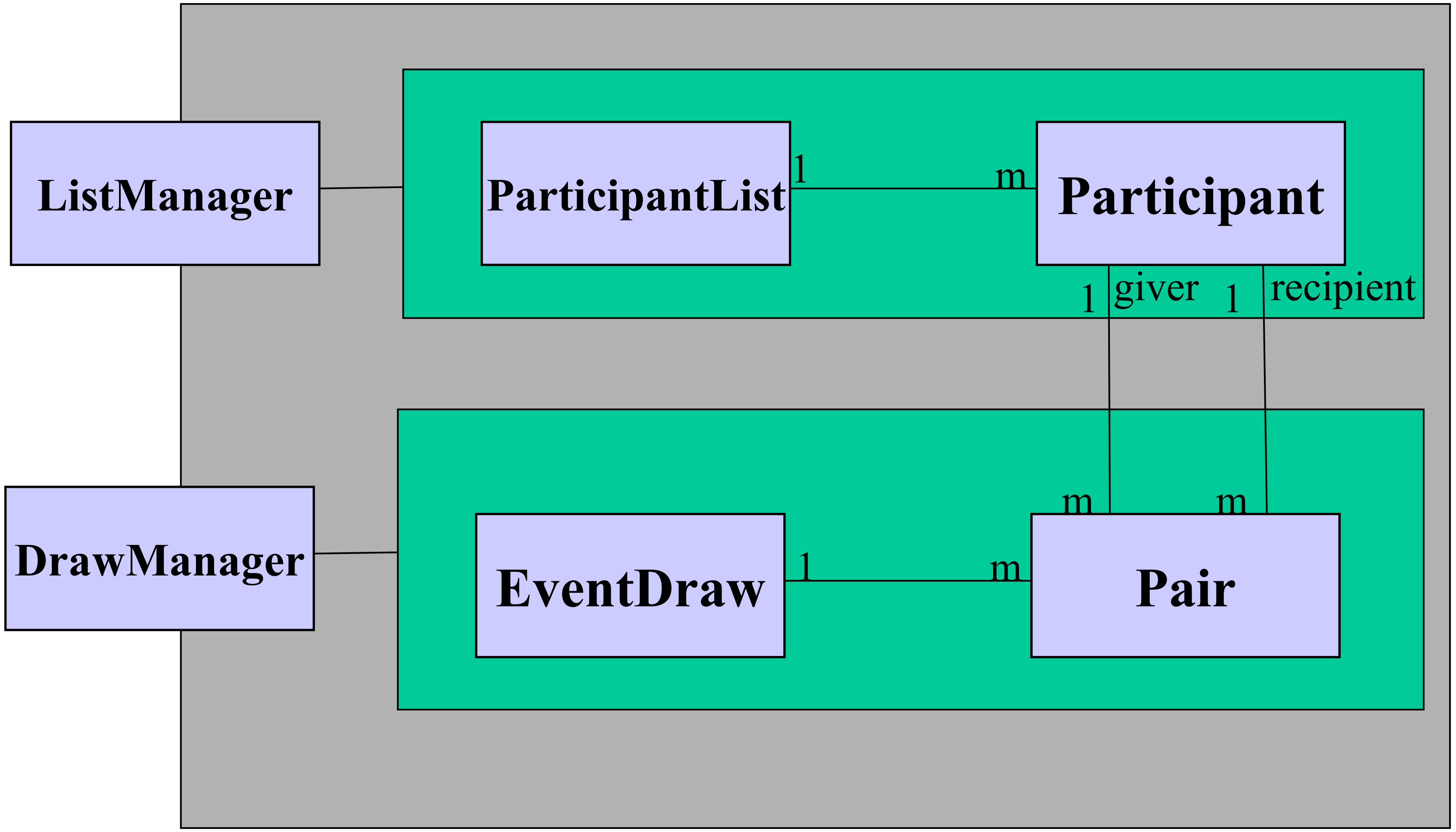
```
@OneToMany(mappedBy="participantList")
public Collection<Participant> getParticipants()
{
    …
```

Relationship Annotations

- @ManyToOne
- @OneToMany

- @OneToOne
- @ManyToMany

Session Facades Are Still in Style



Entity Beans Travel

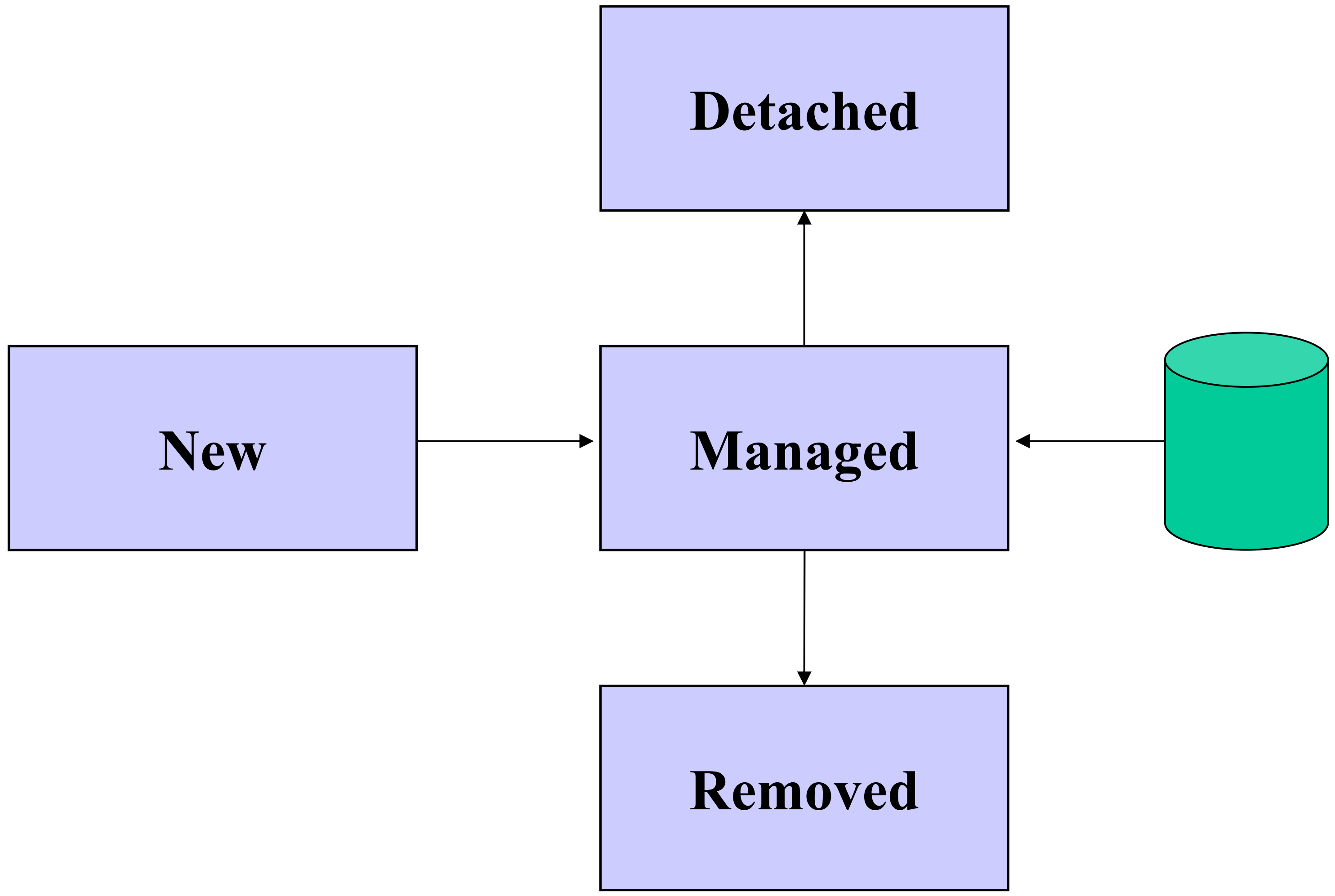
In Client.java:

```
Participant sally = new Participant("Sally", "Johnson",
                                    "sal@priceyisp.com");
```

In Participant.java:

```
@Entity
public class Participant implements java.io.Serializable
{
    …
```

POJO Entity Lifecycle



Merging from the Detached State

In ListManagerBean.java:

```
public ParticipantList addToList(ParticipantList list,
                                 Collection<Participant> newGuys) {
        list = manager.merge(list);
        list.addParticipants( newGuys );
        return list;
}
```

Local and Remote Interfaces

ListManagerBean.java:

```
@Stateless
public class ListManagerBean implements ListManagerRemote,
    ListManagerLocal  {…
```

ListManagerLocal.java:

```
@Local
public interface ListManagerLocal extends ListManager {}
```

ListManagerRemote.java:

```
@Remote
public interface ListManagerRemote extends ListManager {}
```

Some Reflections on the Code

- It’s small and concise – I’m not going to miss ejbActivate and checked exceptions
- …or the redundancy of Data Transfer Objects
- It feels like they’ve given back what I already had from the Java language
- ...and I love the Java 5 language enhancements
- It’s early days but the developer preview releases are already quite useable
- Fun!

Further Information

Specification Drafts:

http://java.sun.com/products/ejb/docs.html

EJB 3 Preview Releases:

http://www.jboss.org/products/ejb3

http://www.oracle.com/technology/tech/java/ejb30.html

http://www.caucho.com/resin-3.0/ejb3/index.xtp

My own thoughts:

http://www.scottcrawford.com/ejb30.html