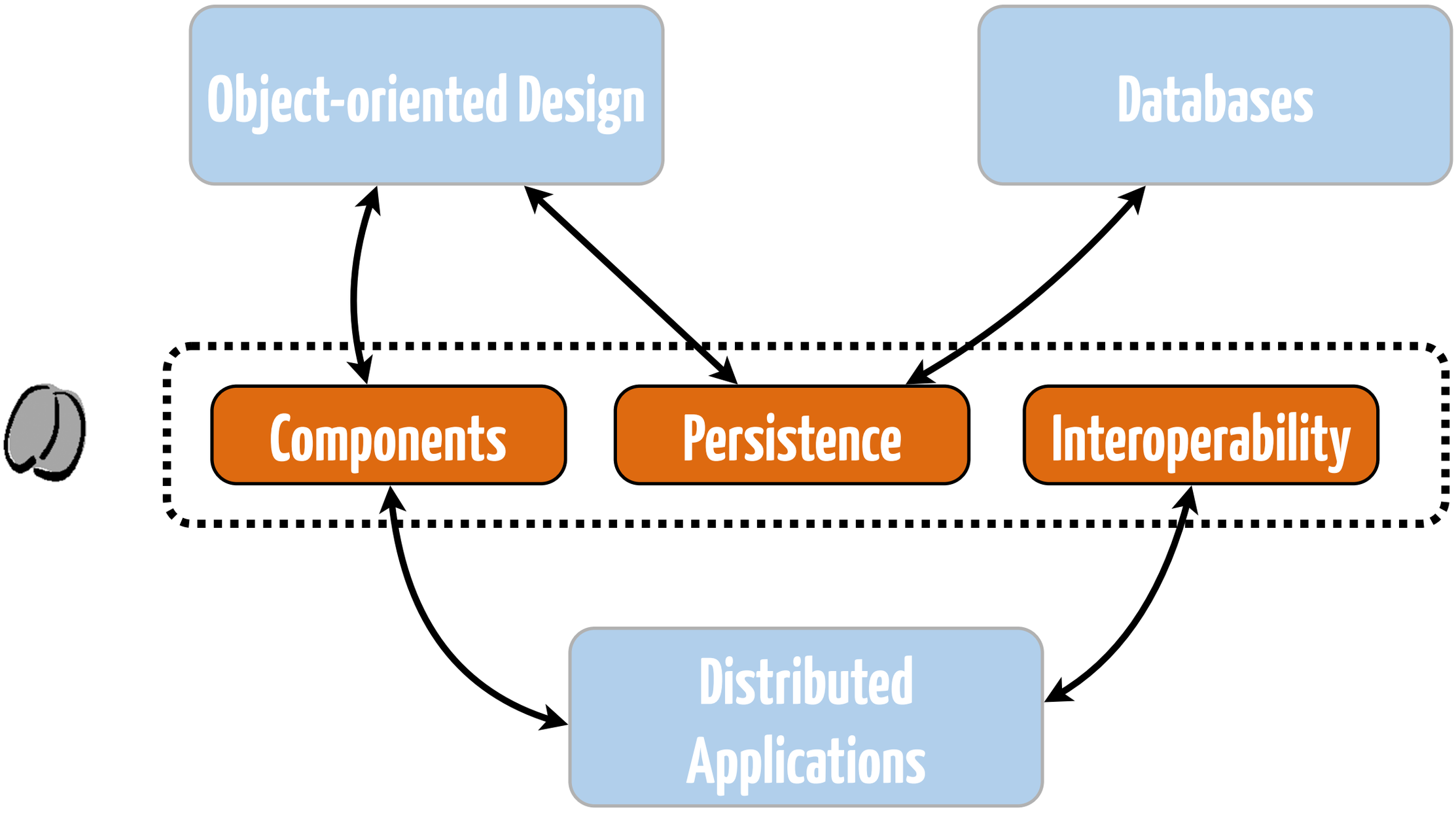
# Enterprise Java Beans 101

Sébastien Mosser

Lecture #2.1, 02.03.2018

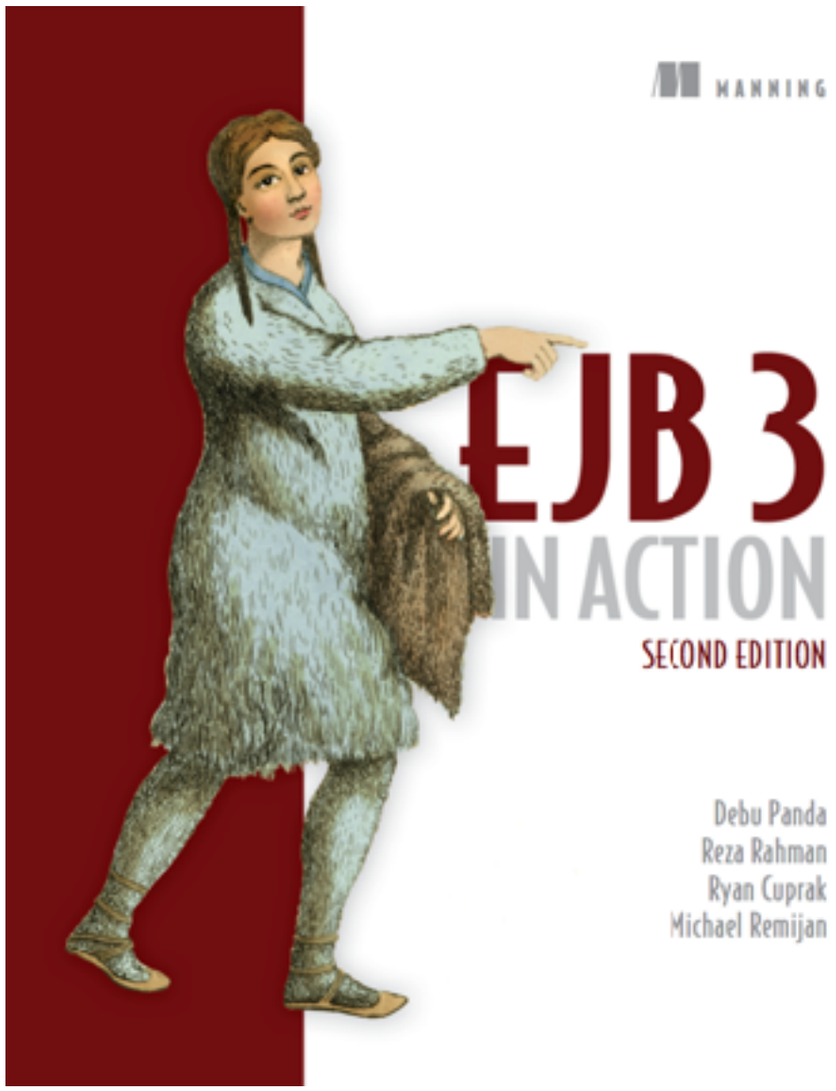
pictures: sxc.hu

# Applications Server: Dependencies



# Bibliography

“ The **ladybug**, the **elephant** and the **cow**




**[EJB3 in Action, 2014]**

# EJB 1 (IBM 1997, Sun 1999)

Do no work

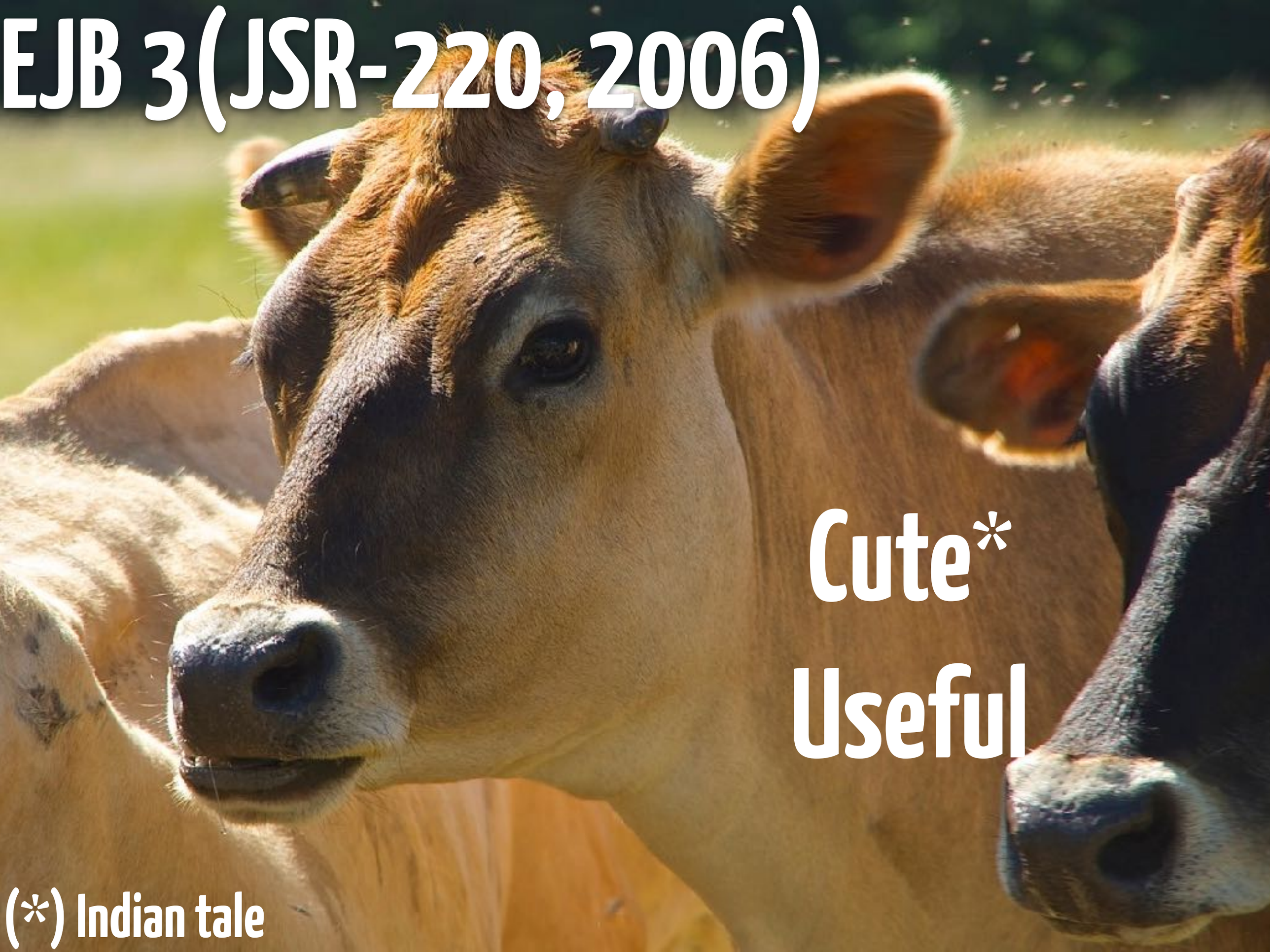
Beautiful

# EJB 2(JSR-19, 2001)

Heavy-weighted

Strong

# EJB 3 (JSR-220, 2006)



Cute*

Useful

(*) Indian tale

1

# Bird's Eye

2

# Inversion of Control

3

# Conclusions

4

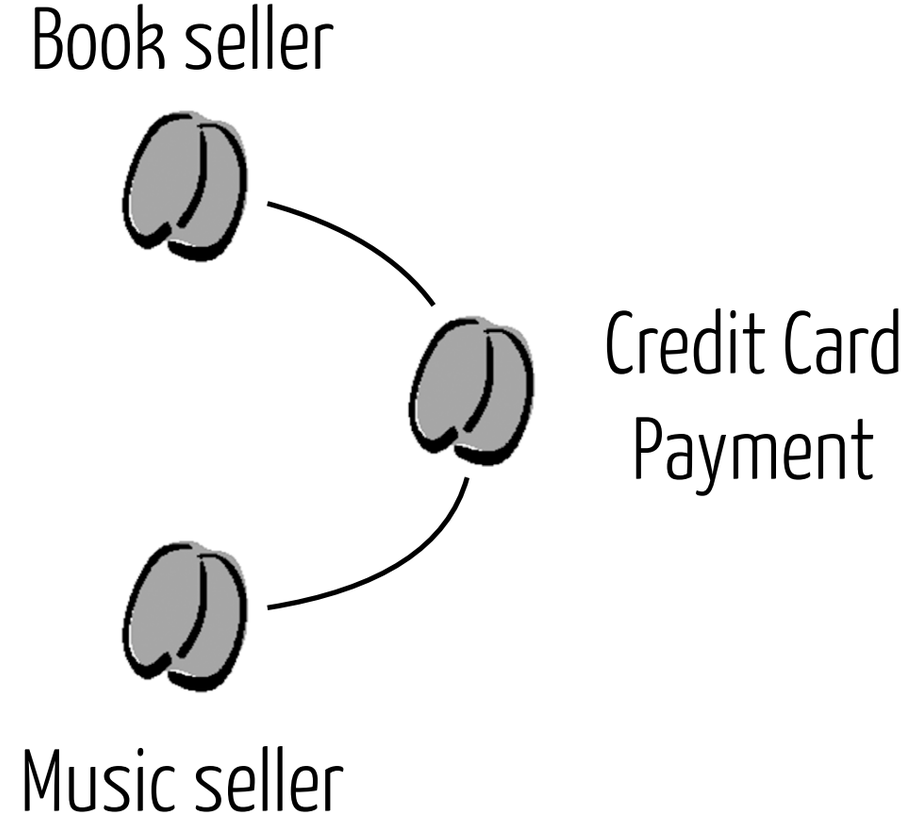
# Bird's Eye

# Overview

# EJB as **Components**

## Interface **reuse**

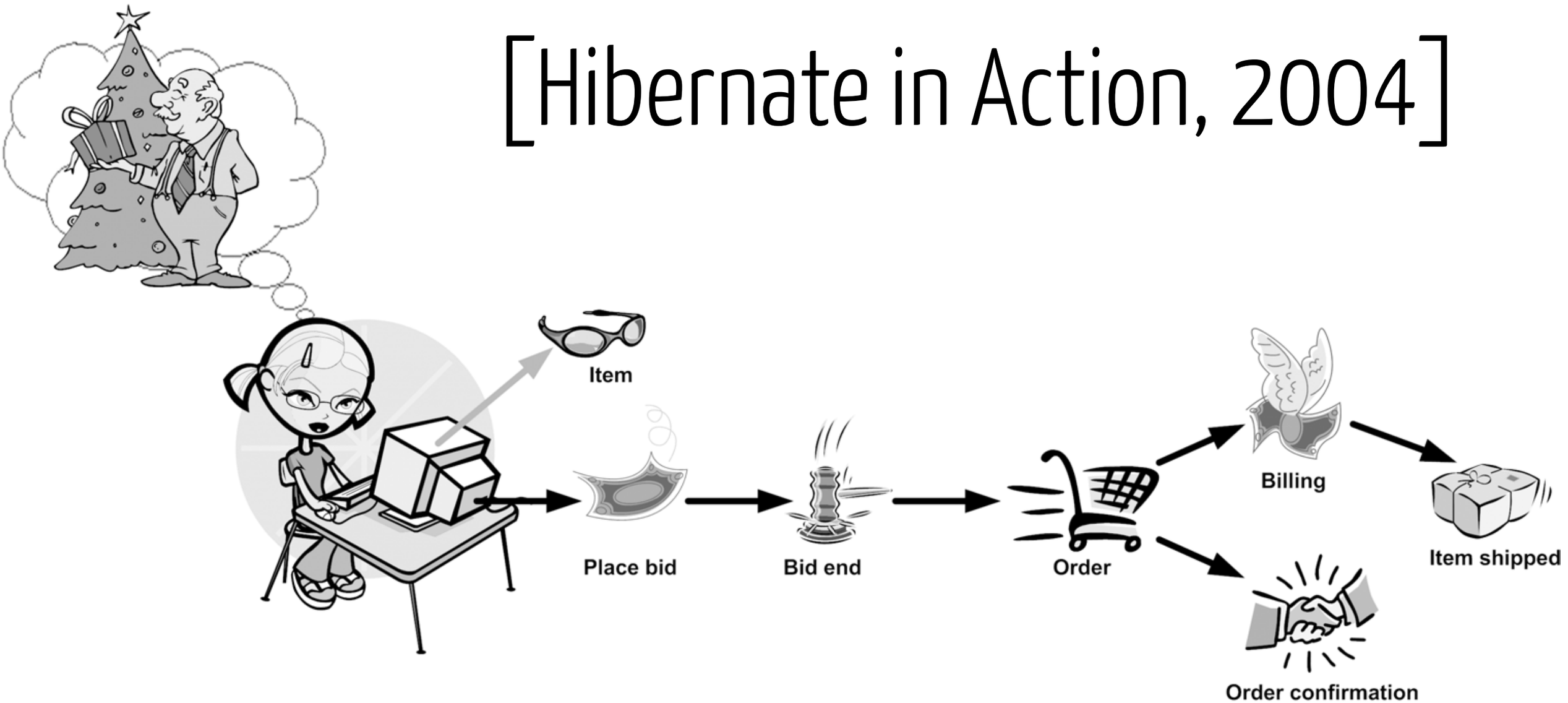
## **Encapsulate** application **Behavior**



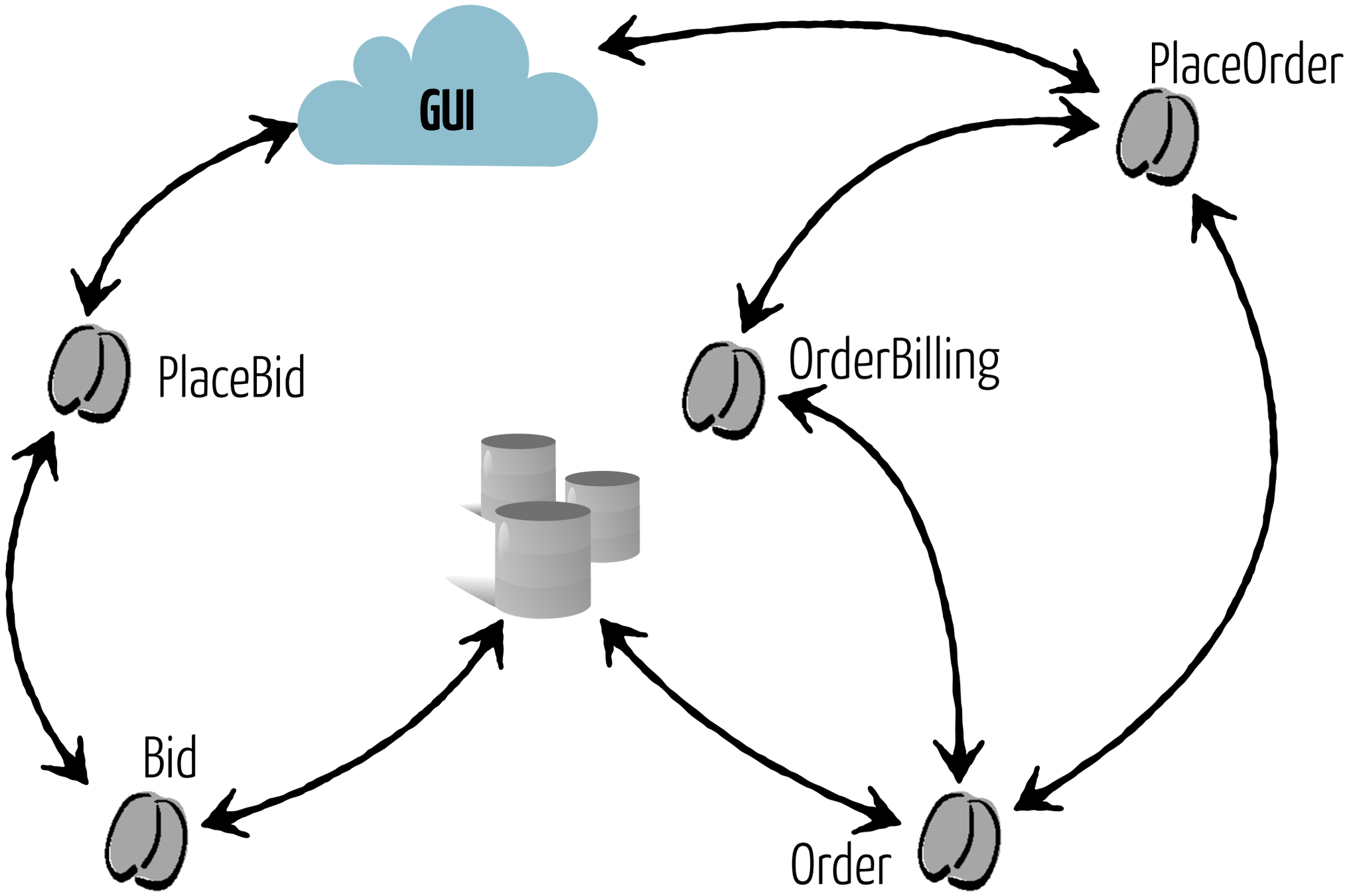

[EiA]

# The ActionBazaar example

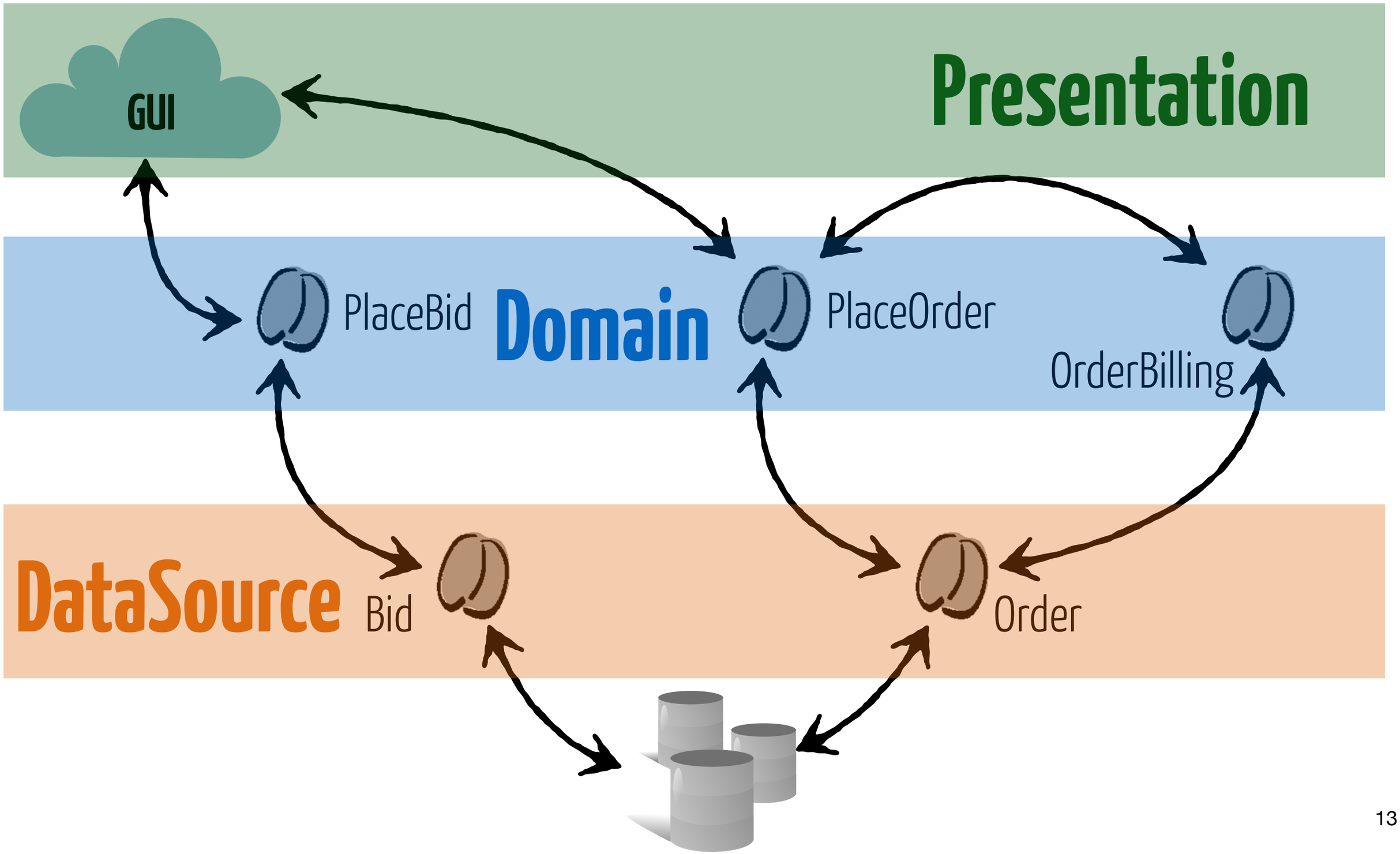
[Hibernate in Action, 2004]



# Identifying **Components**



# 3-tiers Architecture



# Rule of Thumb

**Domain** Beans interfaces as **Verbs**

**DataSource** Beans as **Nouns**

# What the hell is a Bean?

POJO Annotation EJB

EJB = "**Plain Old Java Object** on **Steroids**"

[EiA]

# POJO on Steroids

```
public interface HelloUser {
   public void sayHello(String n);
}
```

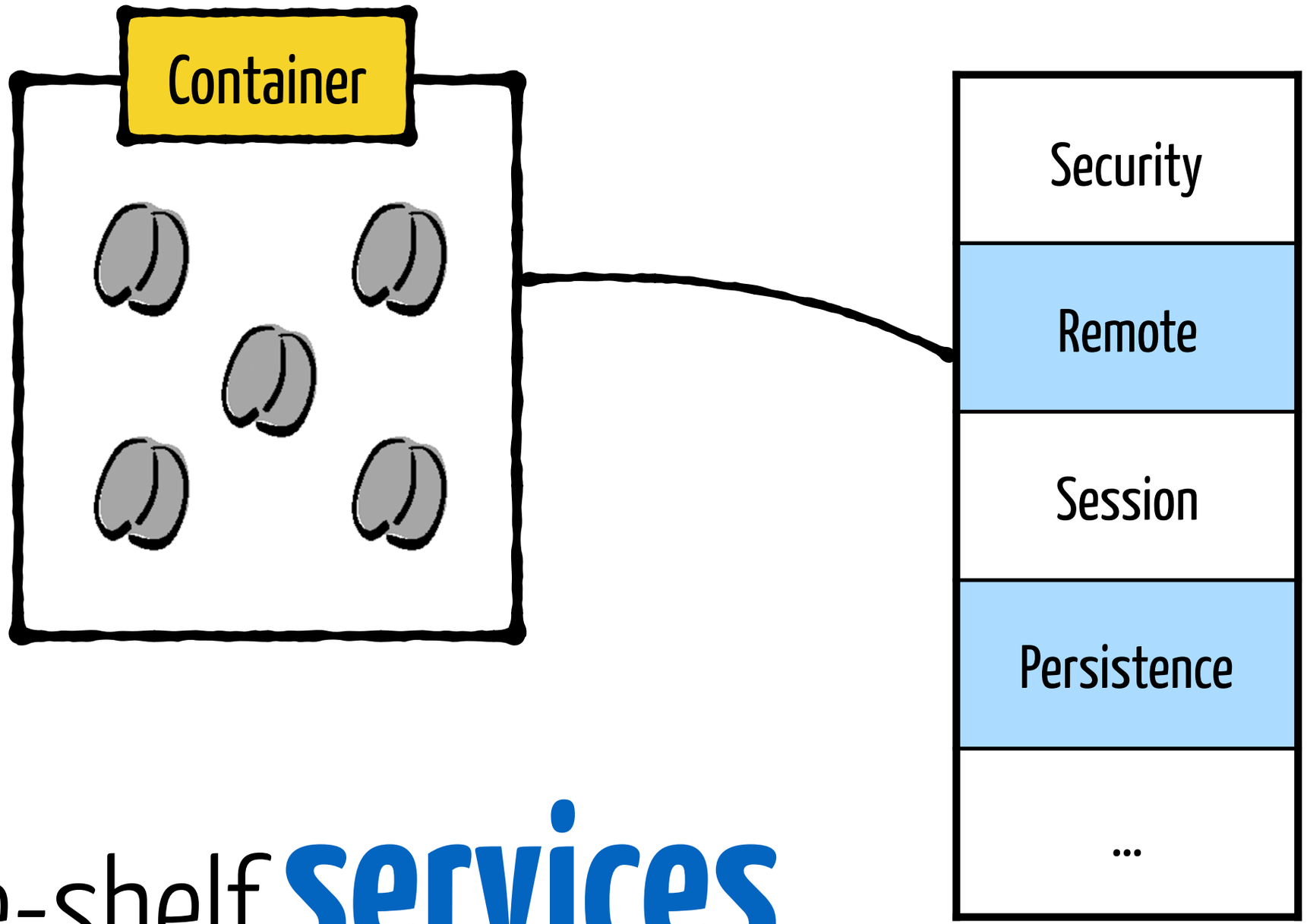
```
public class HelloUserBean implements HelloUser{
   public void sayHello(String n) {
     System.out.println("Hello, " + n "!");
   }
}
```

# POJO on Steroids

```
public interface HelloUser {
   public void sayHello(String n);
}
```

```
@javax.ejb.Stateless
public class HelloUserBean implements HelloUser{
   public void sayHello(String n) {
      System.out.println("Hello, " + n "!");
   }
}
```

# Steroids? EJB as a **Framework**




Off-the-shelf **services**

# Different **Flavors**

**Domain**

Session Bean

Message-driven Bean

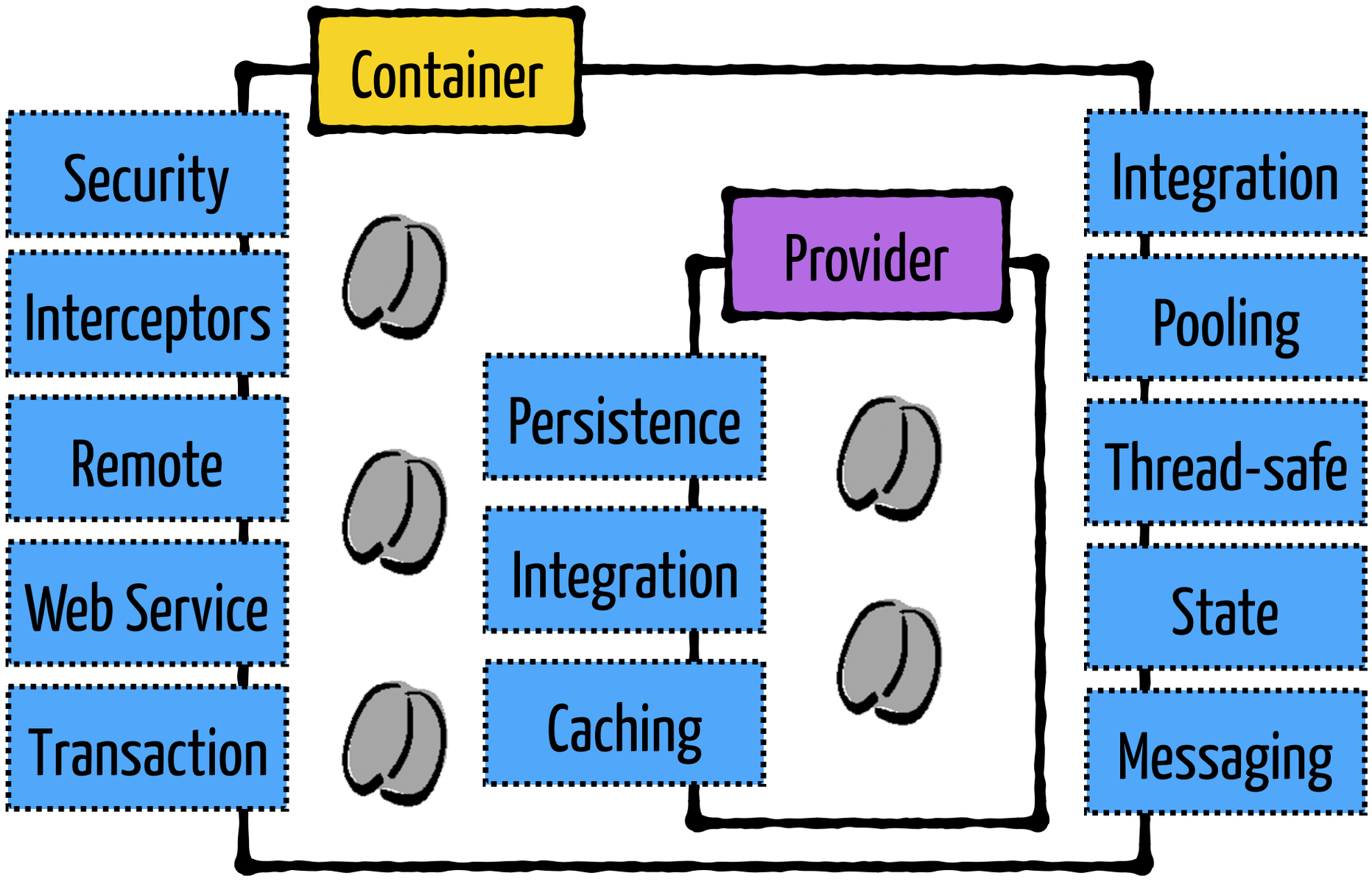
**DataSource**

Entity Bean

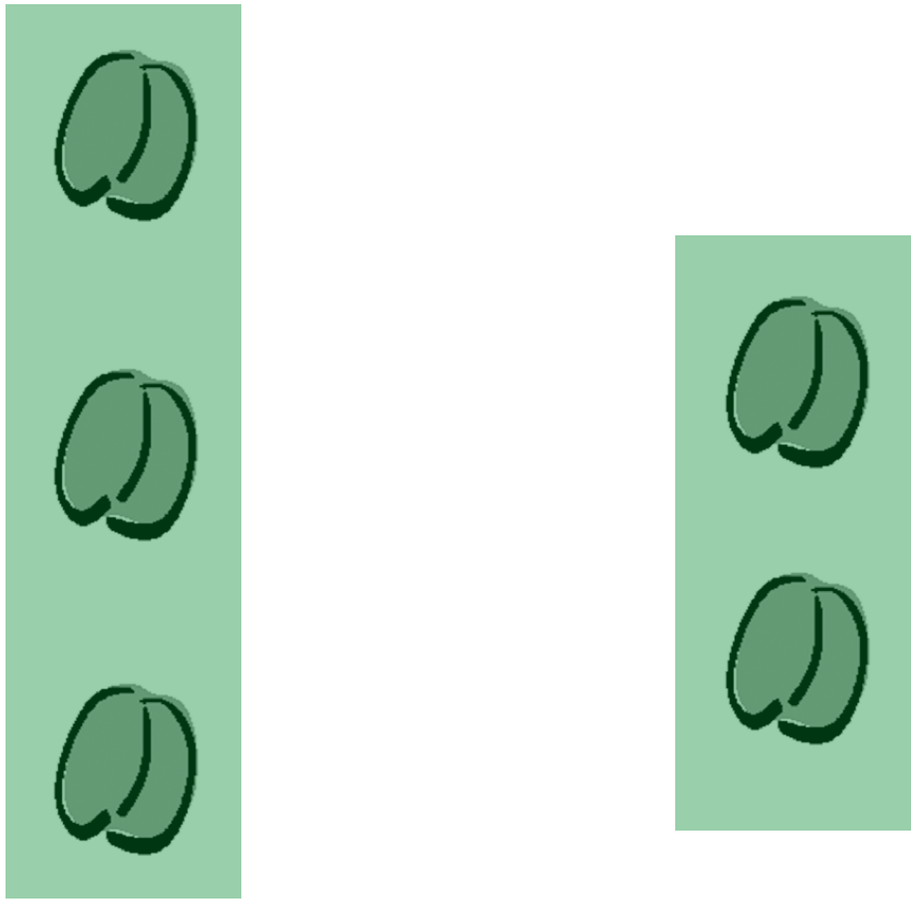
**Container**

**Provider**

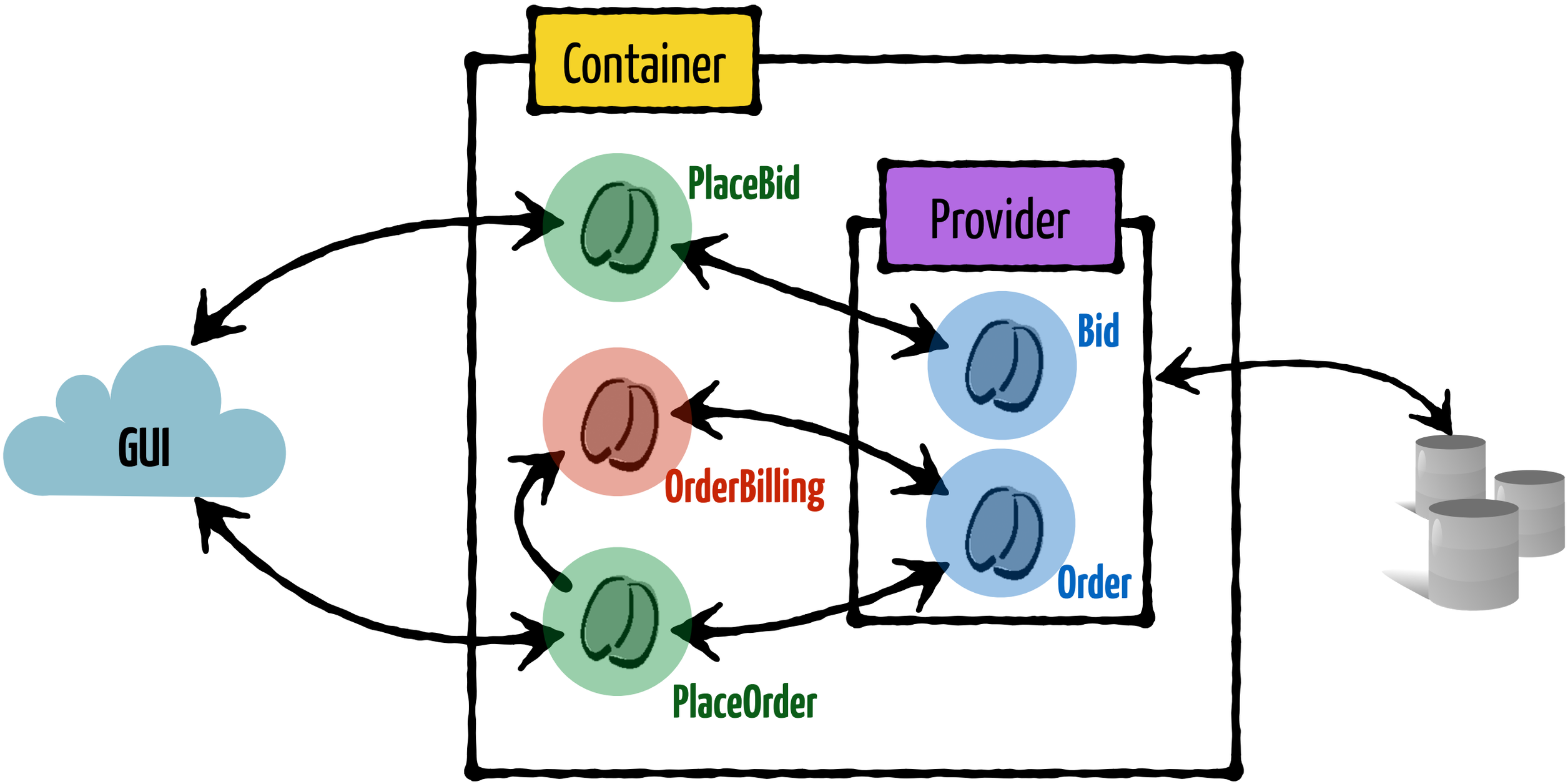
# EJBs **Services**: focus on **Business**!



# EJBs **Services**: focus on **Business**!



# Session, Message-driven & Entity



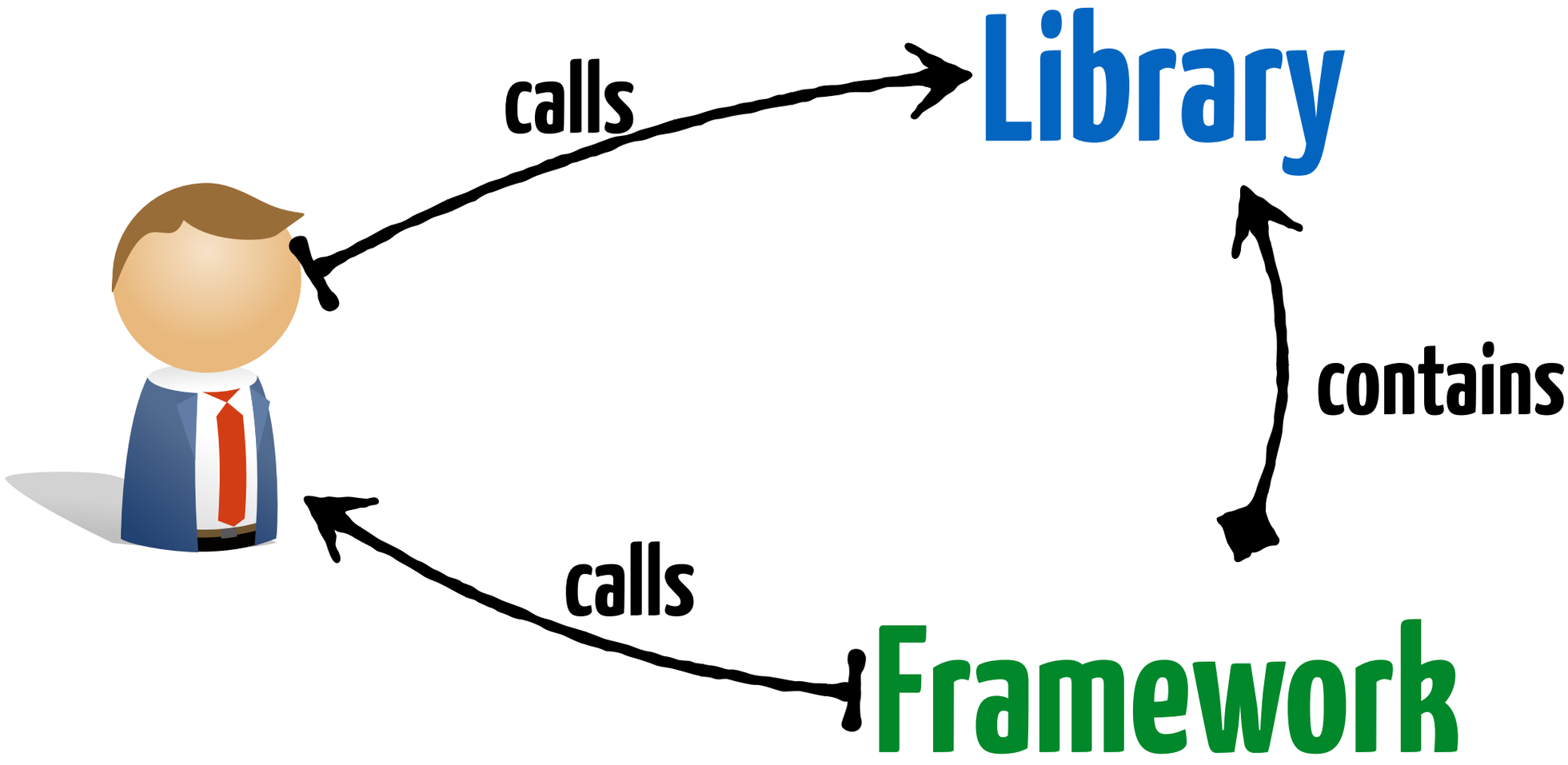
# Inversion of Control

# Library versus Framework?

## Library

## Framework

# Library versus Framework?



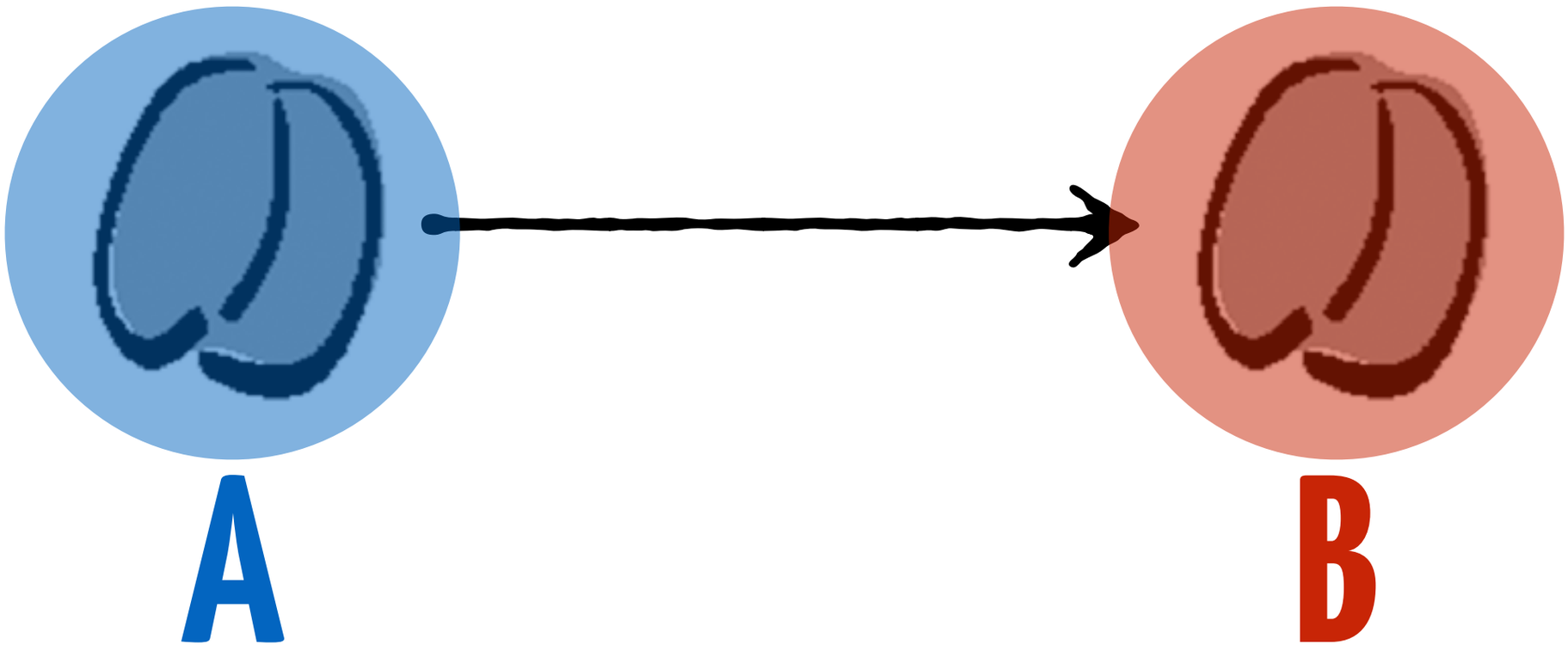
# Inversion of Control

Your **code** reuses a **library**

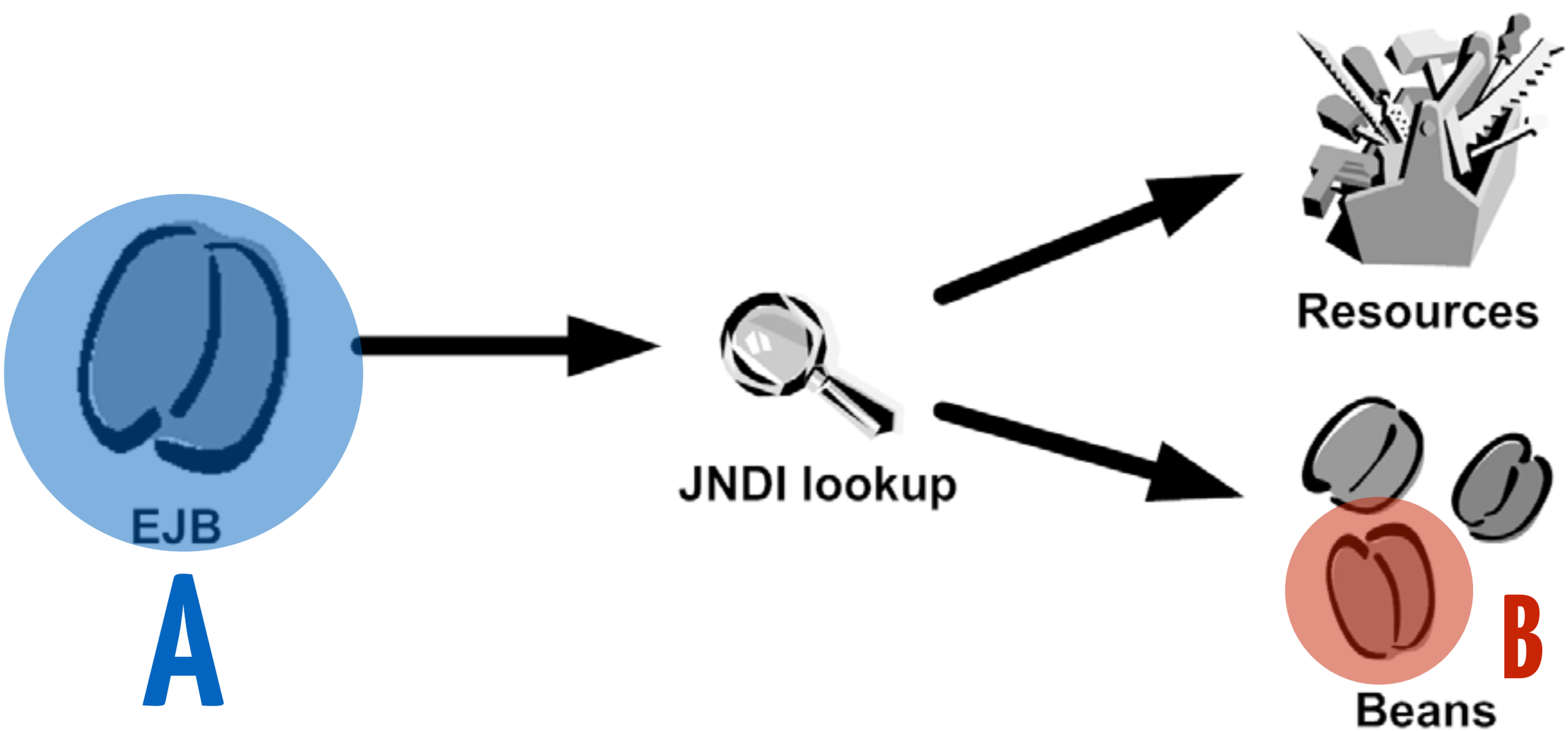
A **framework** reuses your **code**


HOTEL

# Problem: Bean Dependencies

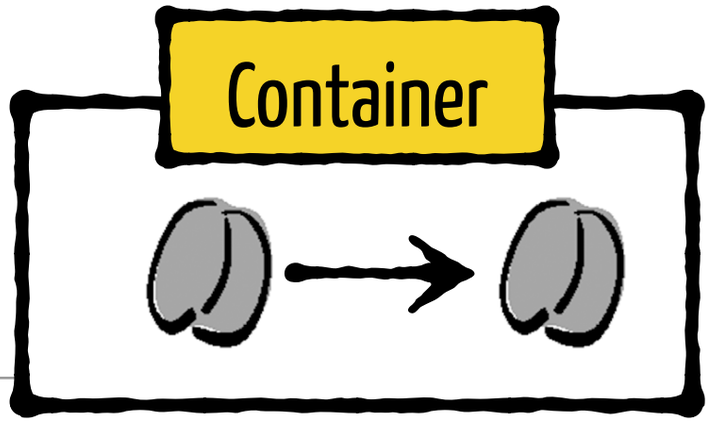


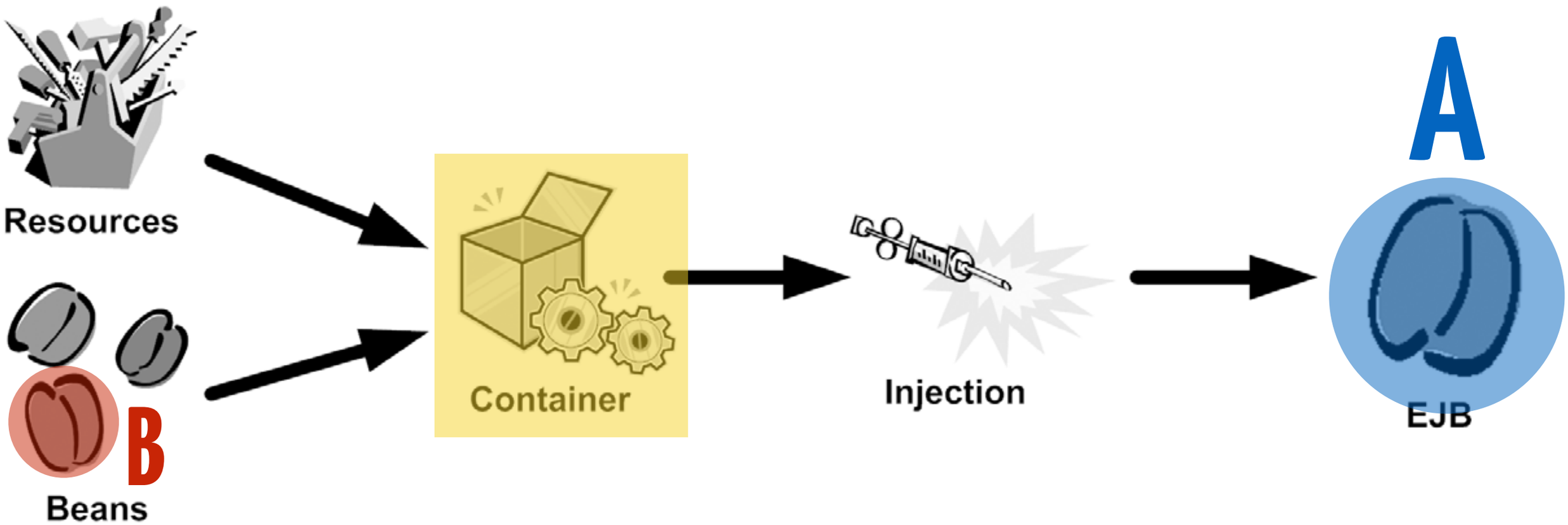
# The good-old method: Lookup




```
Object ejbHome = new InitialContext().lookup("java:comp/env/PlaceBid");
PlaceBidHome placeBidHome = (PlaceBidHome)
    PortableRemoteObject.narrow(ejbHome, PlaceBidHome.class);
PlaceBid placeBid = placeBidHome.create();
```

# Steroids: Dependency Injection




```
public class PlaceOrderTestClient {
    @EJB
    private static PlaceOrder placeOrder;
```

**Injects an instance of EJB**

# Conclusions

3

# What the hell is a Bean?

POJO Annotation EJB

EJB = "**Plain Old Java Object** on **Steroids**"

[EiA]

# Different **Flavors**

**Domain**

Session Bean

Message-driven Bean

**DataSource**

Entity Bean

# EJBs **Services**: focus on **Business**!