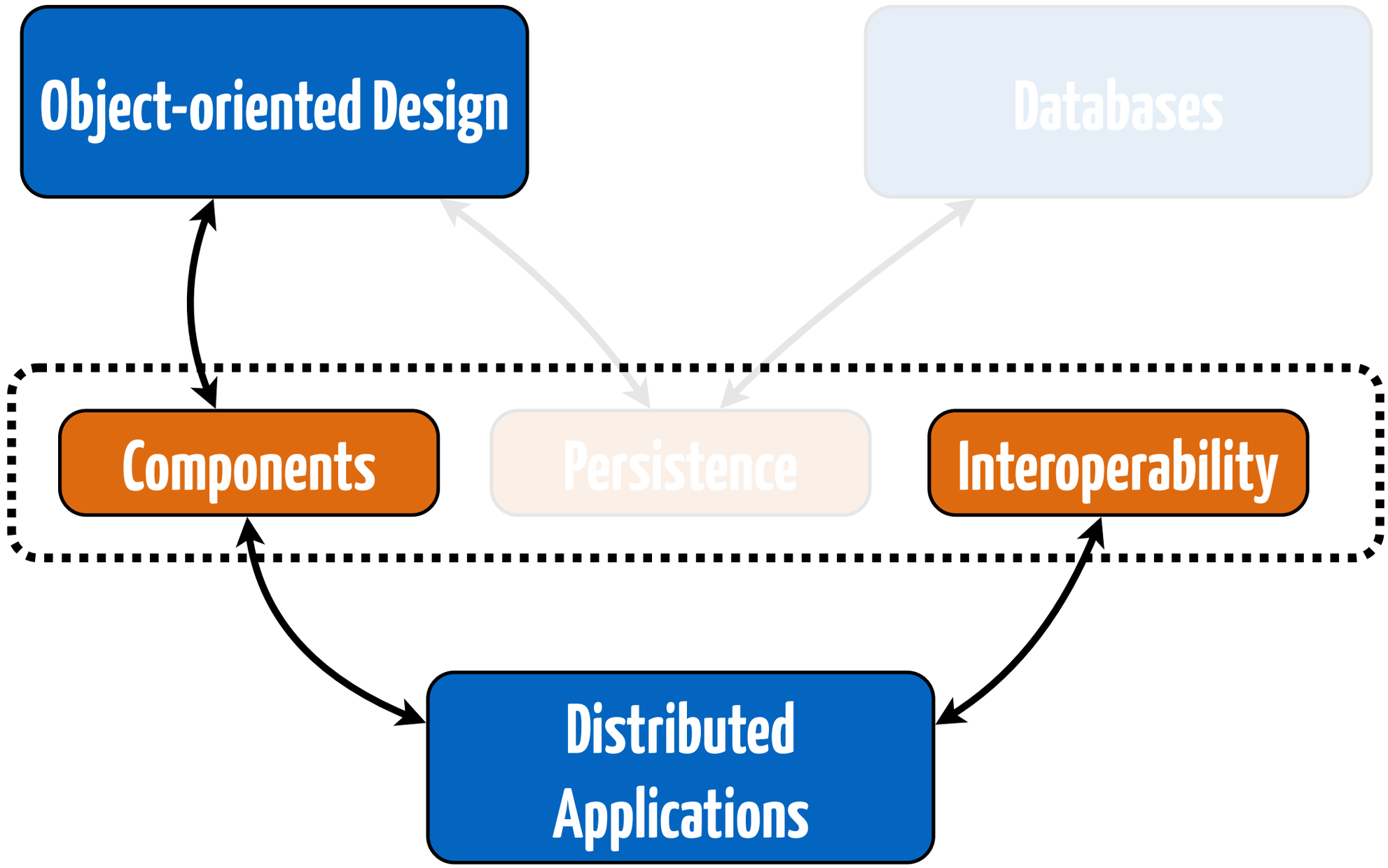
Session Beans 101

Applications Server: Dependencies



3-tiers architecture

Presentation

Handle users

Domain

Business functionality

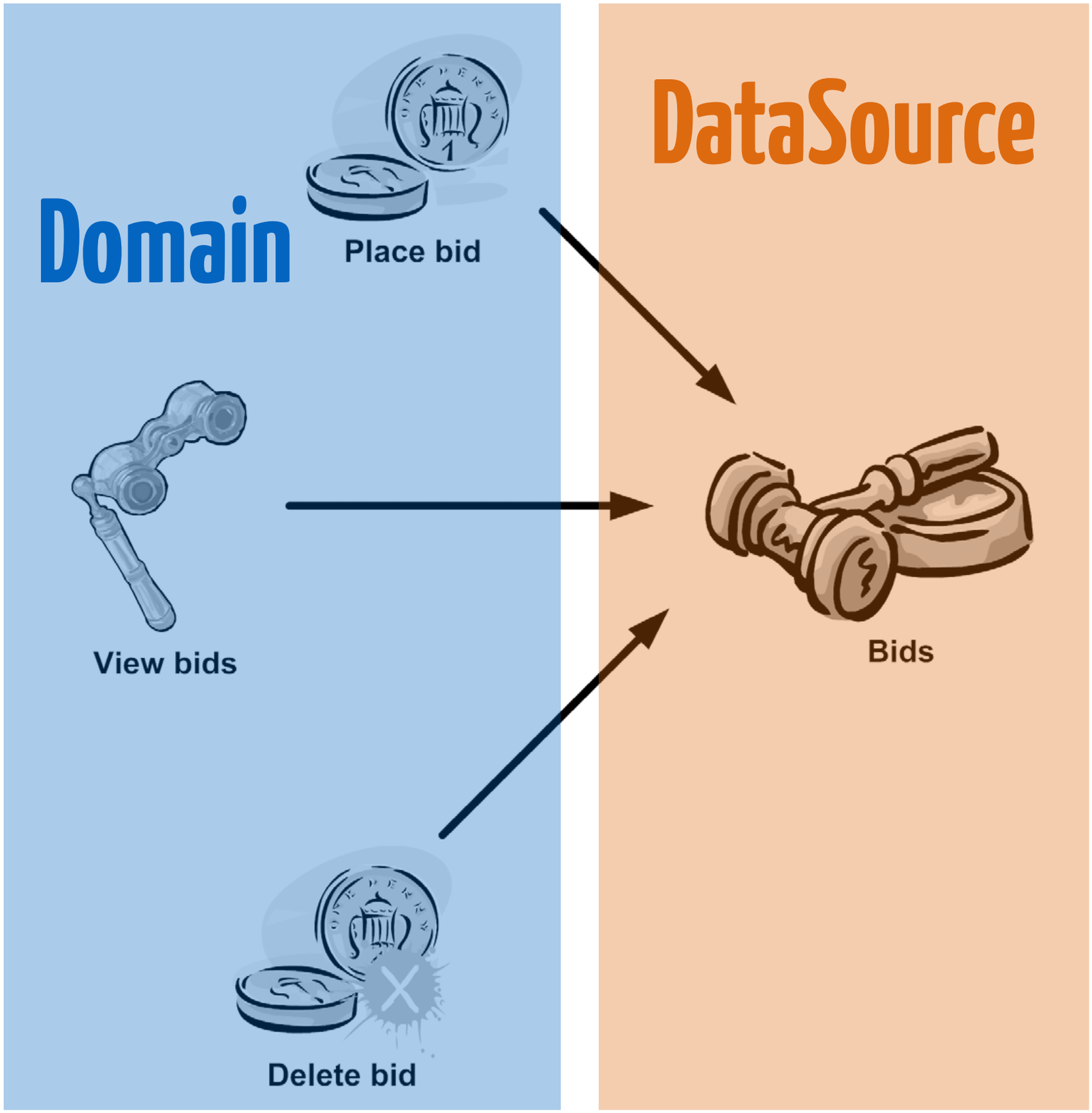
Data Source

Handle data storage

Rule of Thumb

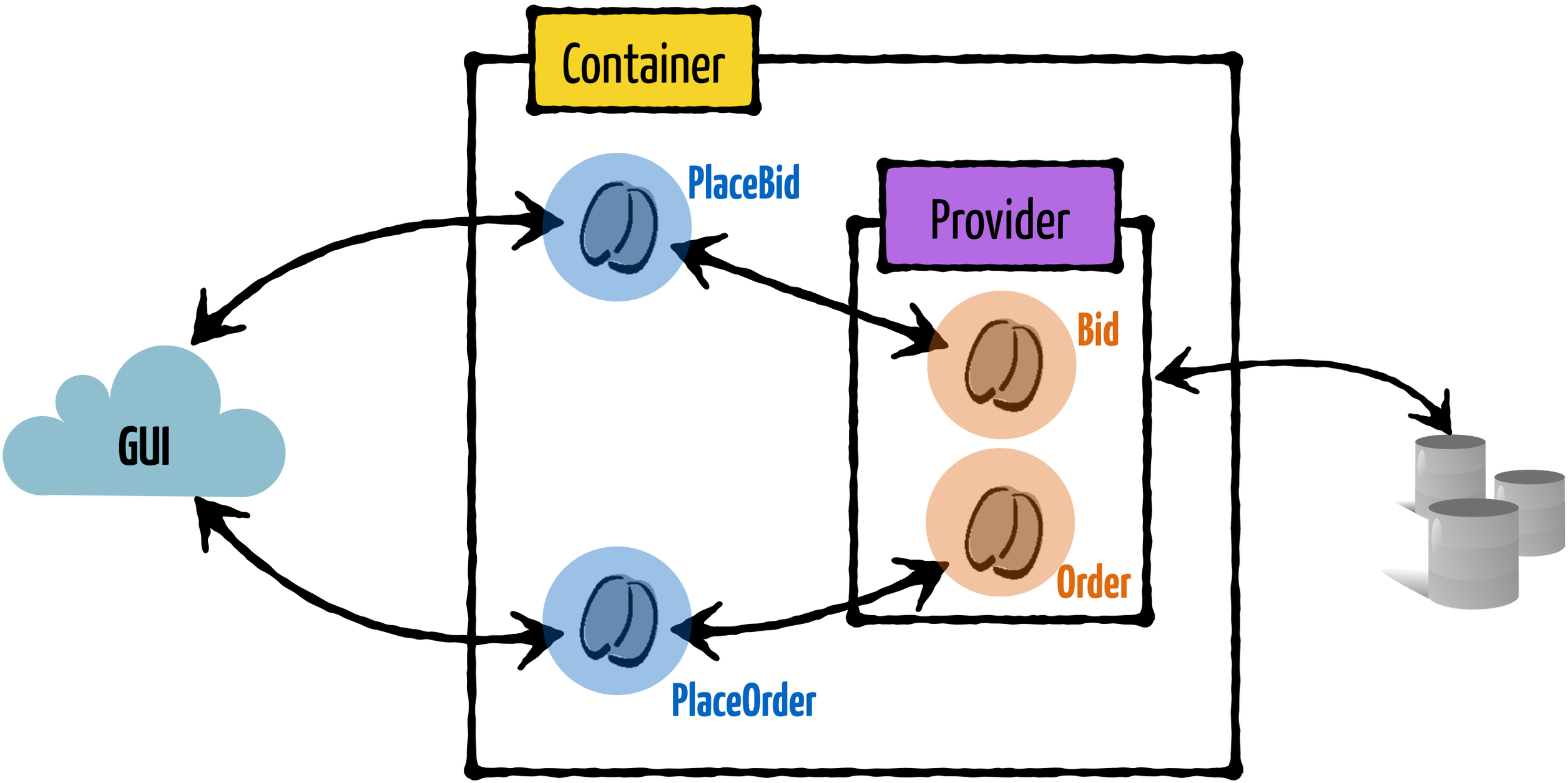
Domain Bean interfaces as **Verbs**

DataSource Beans as **Nouns**



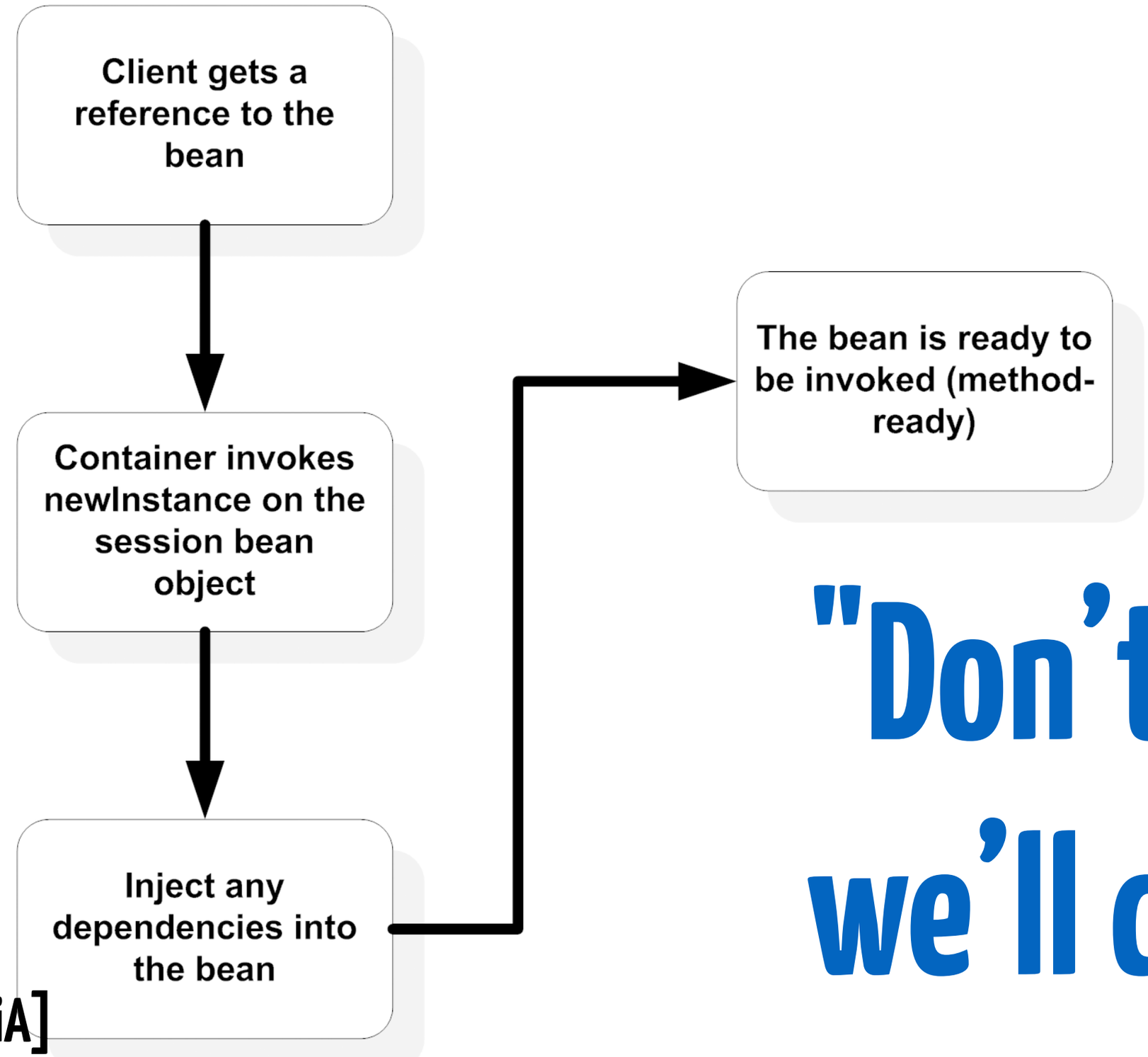
[EiA]

Client **never calls** a datasource directly



You'll **never** instantiate a domain bean.

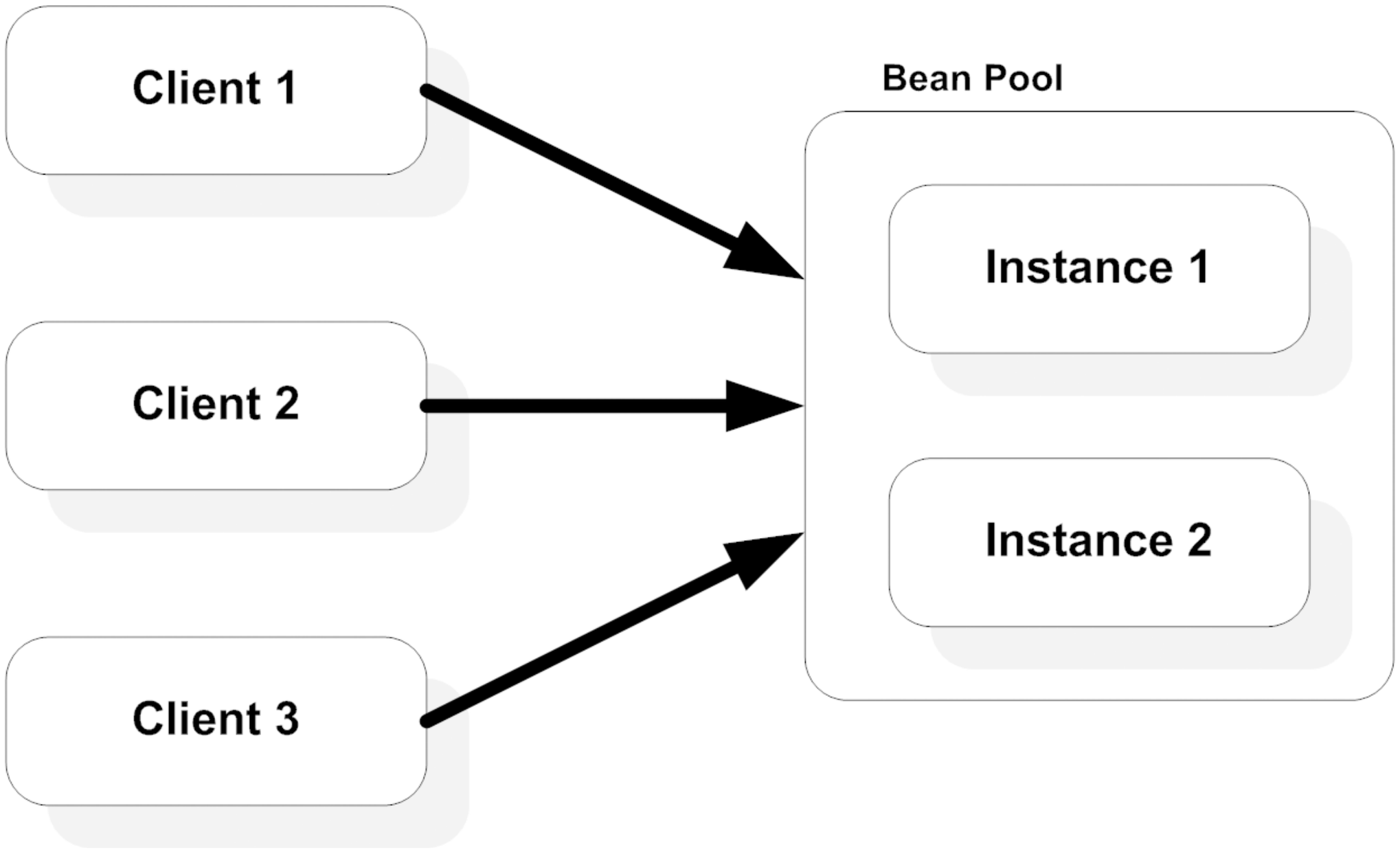
EJB's Lifecycle: **Inversion of Control**



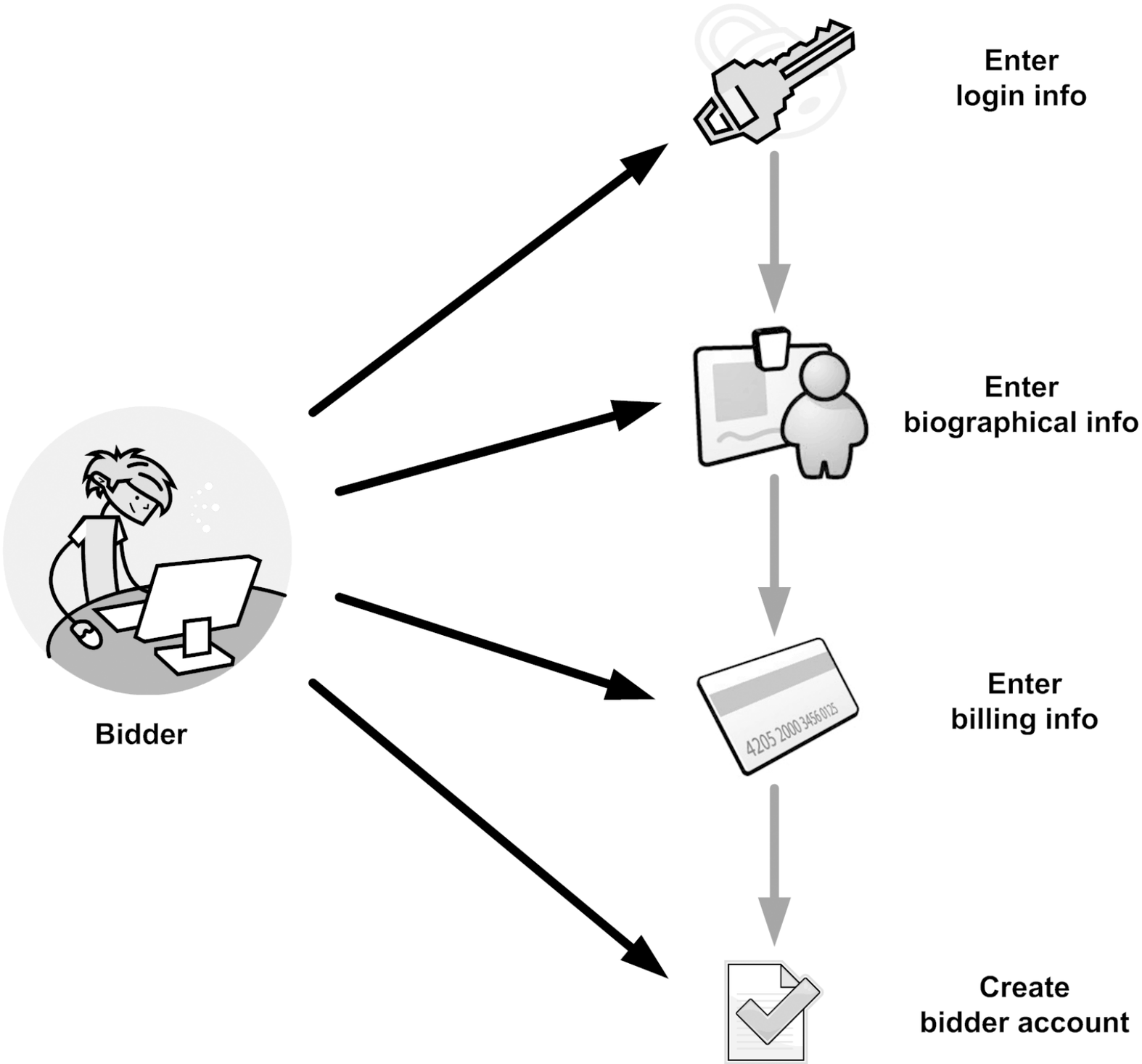
"Don't call us, we'll call you"

[EiA]

EJBs live in a "**pool**"



Domain beans live during a **Session**



state ful/less beans

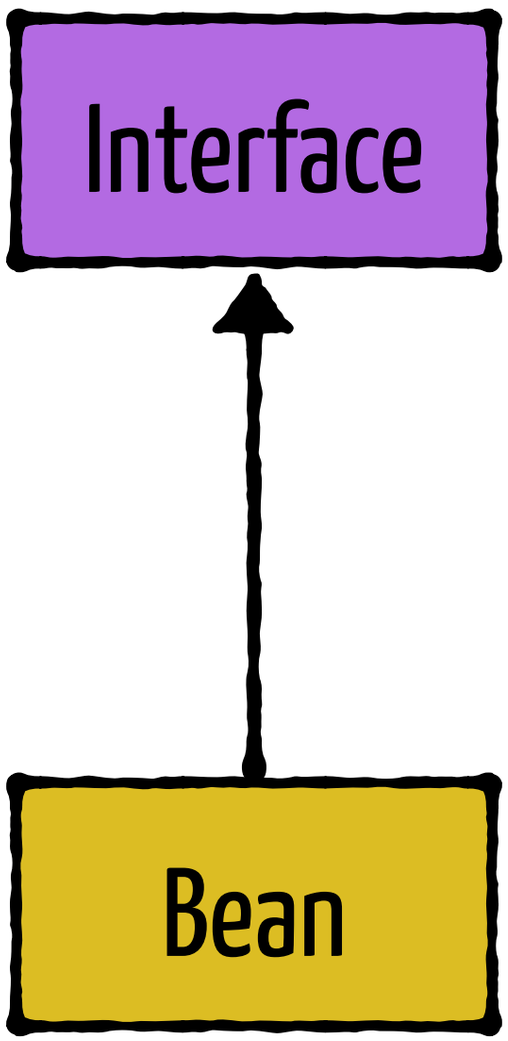
refer to the

conversational state

during a session

State(less|ful) beans | 101

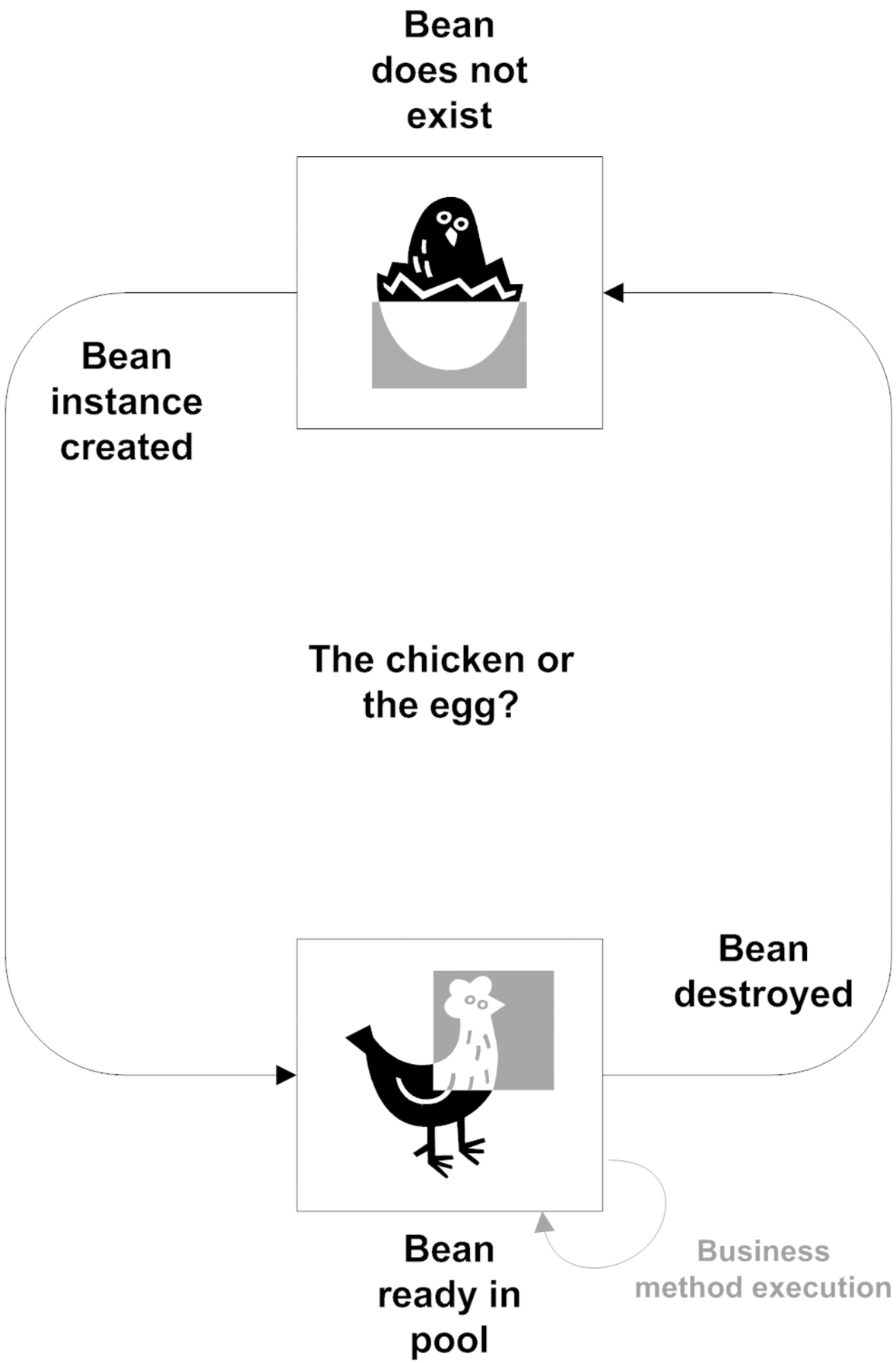
Stateless beans : POJO + Annotations




```
public interface PetManager {
    public Pet create(String name);
}
```

```
@Stateless
public class PetManagerBean implements PetManager {

    @PersistenceContext
    EntityManager entityManager;

    @Override
    public Pet create(String name){
        Pet p = new Pet(name);
        entityManager.persist(p);
        return p;
    }

}
```

Lifecycle



Handled by the container

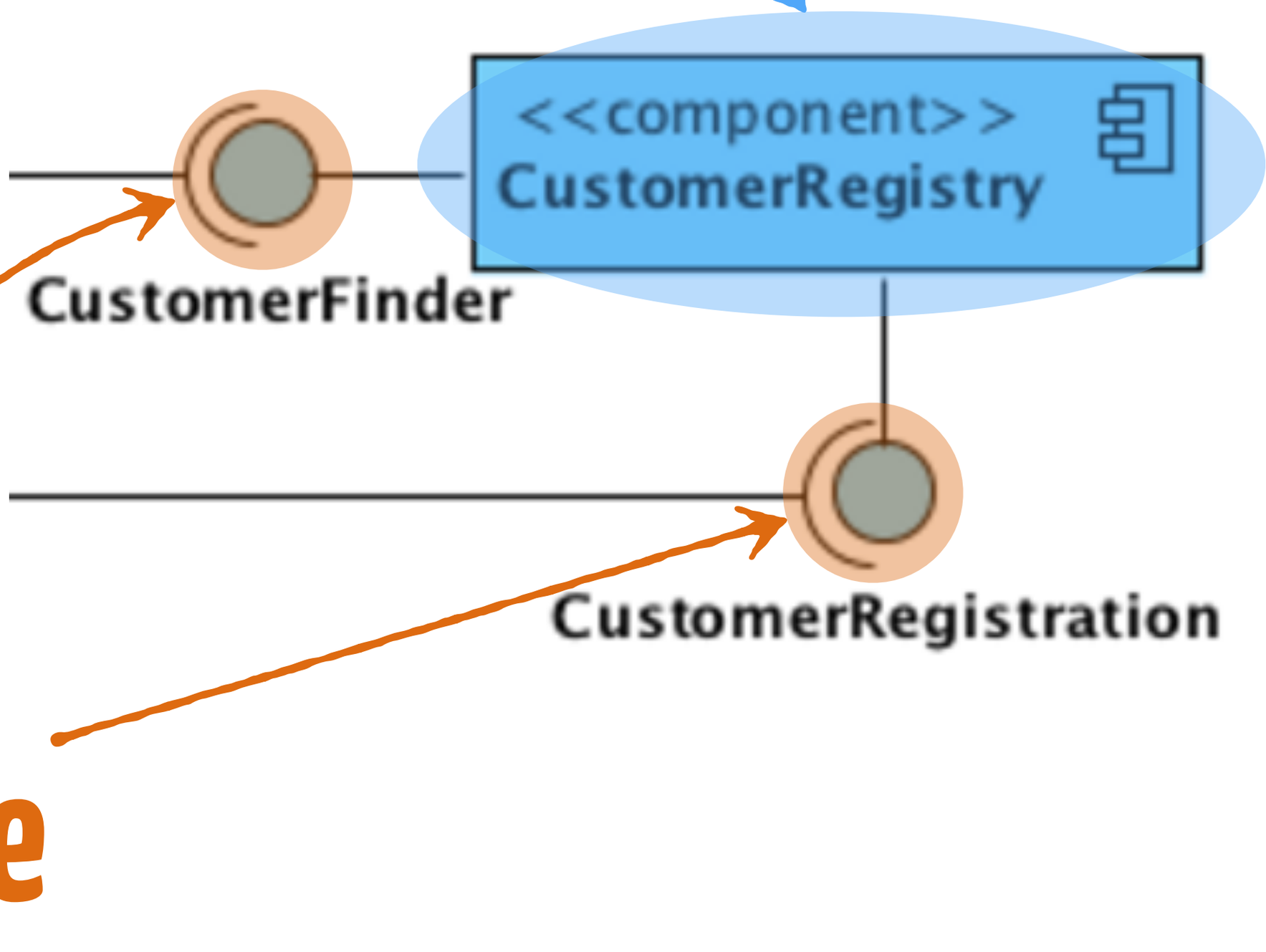
Consuming a Bean: Inversion of Control

```
@EJB
private PetManager manager;
```

Interface

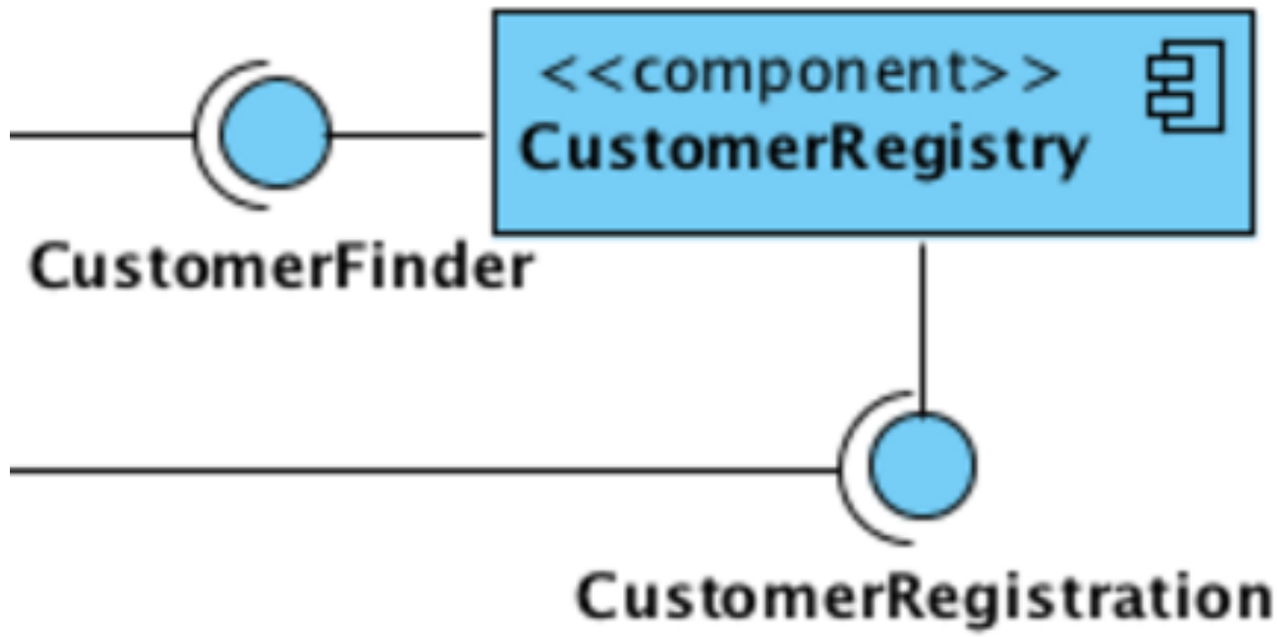
```
@Test
public void testCreation() throws Exception {
   Pet jinx = manager.create("Jinx");
   assertEquals(jinx.name, "Jinx");
}
```

Stateless Bean



Interface

```java
@Local
public interface CustomerFinder {

    Optional<Customer> findByName(String name);

}
```

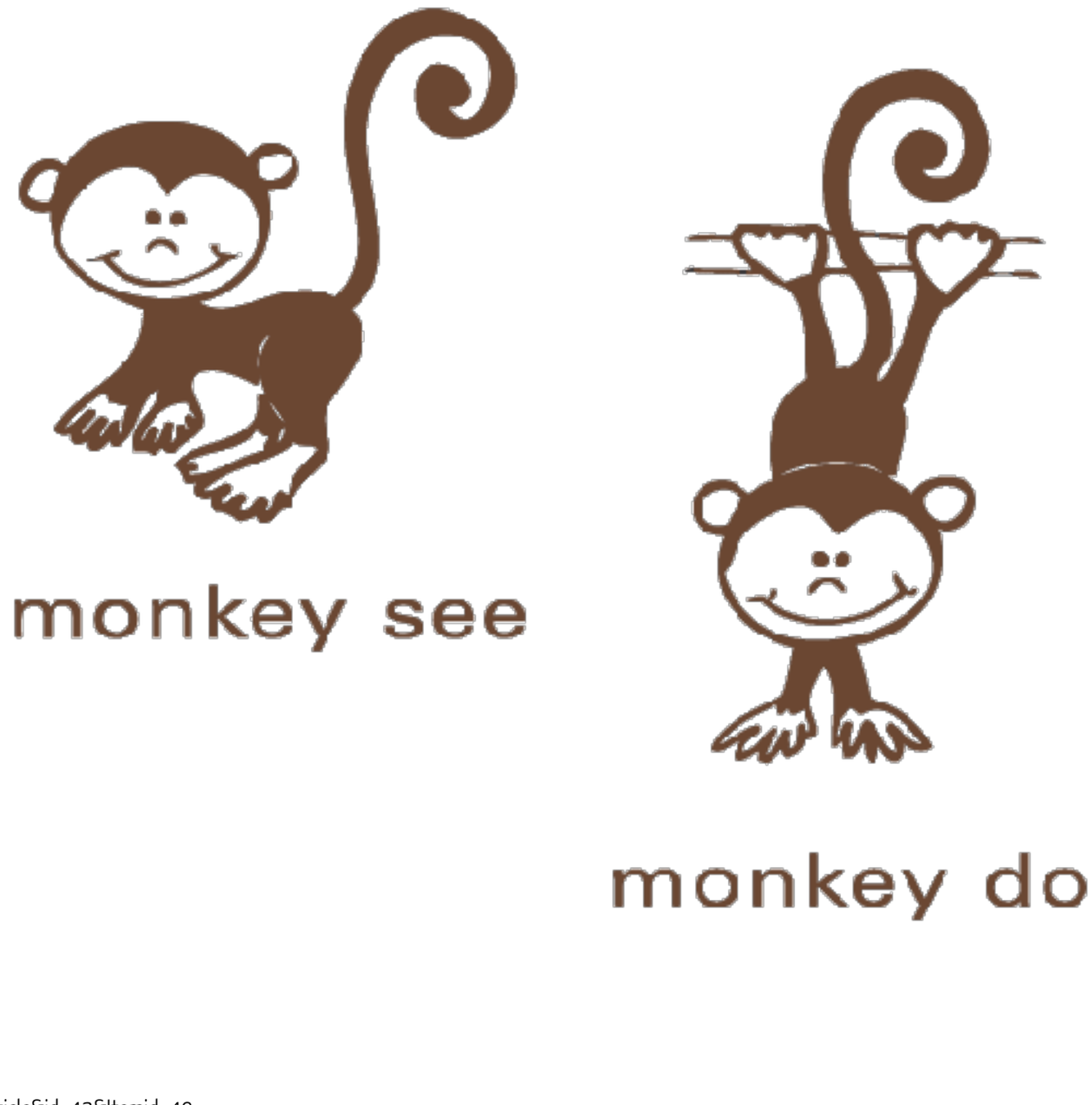



```java
@Local
public interface CustomerRegistration {

    void register(String name, String creditCard)
            throws AlreadyExistingCustomerException;

}
```

```java
@Stateless
public class CustomerRegistryBean
        implements CustomerRegistration, CustomerFinder {

    @EJB
    private Database memory;

    /**********************************************
     ** Customer Registration implementation **
     **********************************************/

    @Override
    public void register(String name, String creditCard)
            throws AlreadyExistingCustomerException {
        if(findByName(name).isPresent())
            throw new AlreadyExistingCustomerException(name);
        memory.getCustomers().put(name, new Customer(name, creditCard));
    }


    /**********************************************
     ** Customer Finder implementation **
     **********************************************/

    @Override
    public Optional<Customer> findByName(String name) {
        if (memory.getCustomers().containsKey(name))
            return Optional.of(memory.getCustomers().get(name));
        else
            return Optional.empty();
    }

}
```

Persistence mock

EJB are **standard**: Learn by **Example**!



http://infiltrated.net/index.php?option=com_content&view=article&id=43&Itemid=49

Want to learn more on state and beans ?

https://github.com/collet/4A_ISA_TheCookieFactory/blob/develop/chapters/BusinessComponents.md

First

More on a specific course later...