Persistence - Part 1

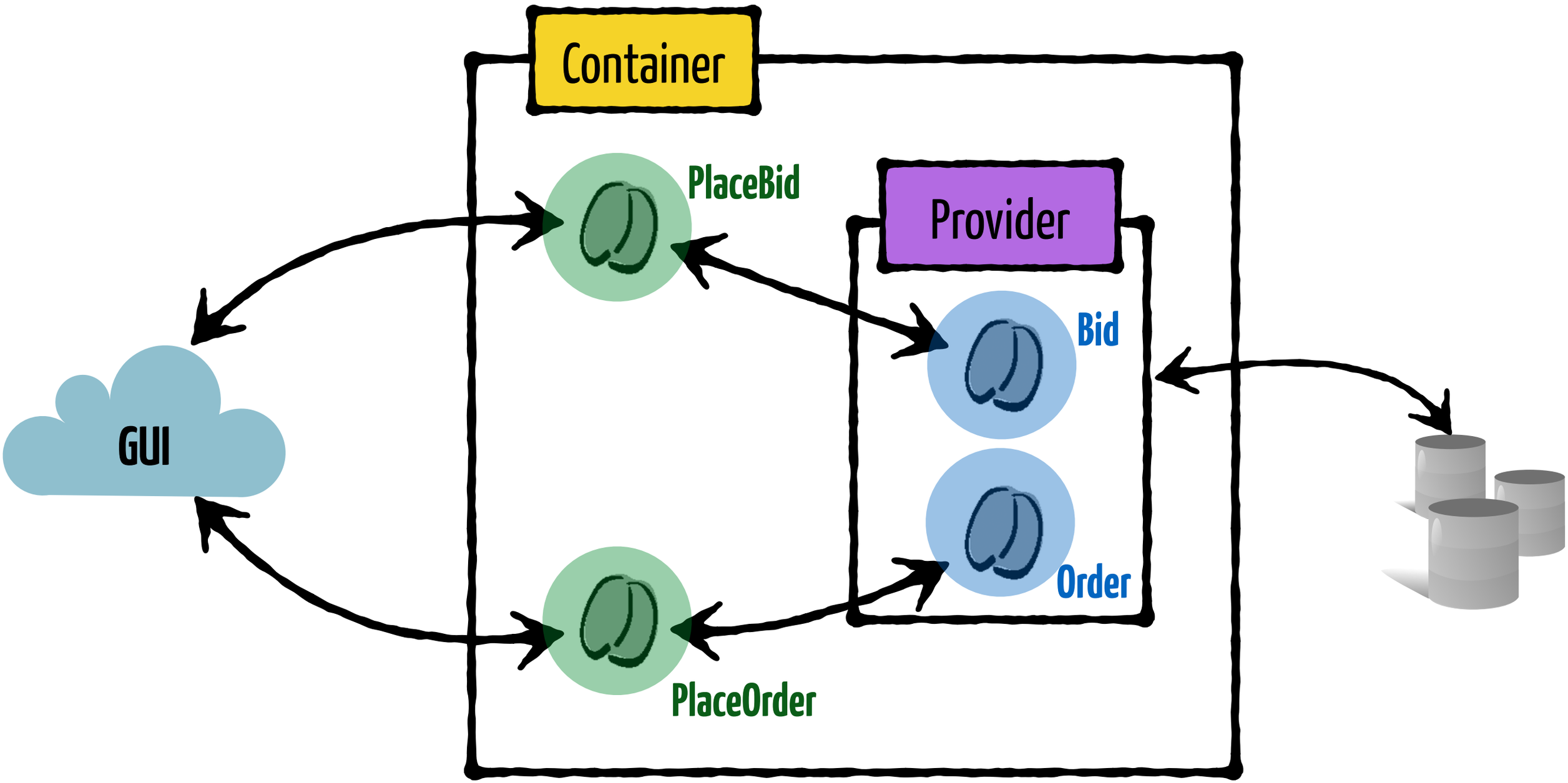
Philippe Collet, contains 78,3% of slides from Sébastien Mosser

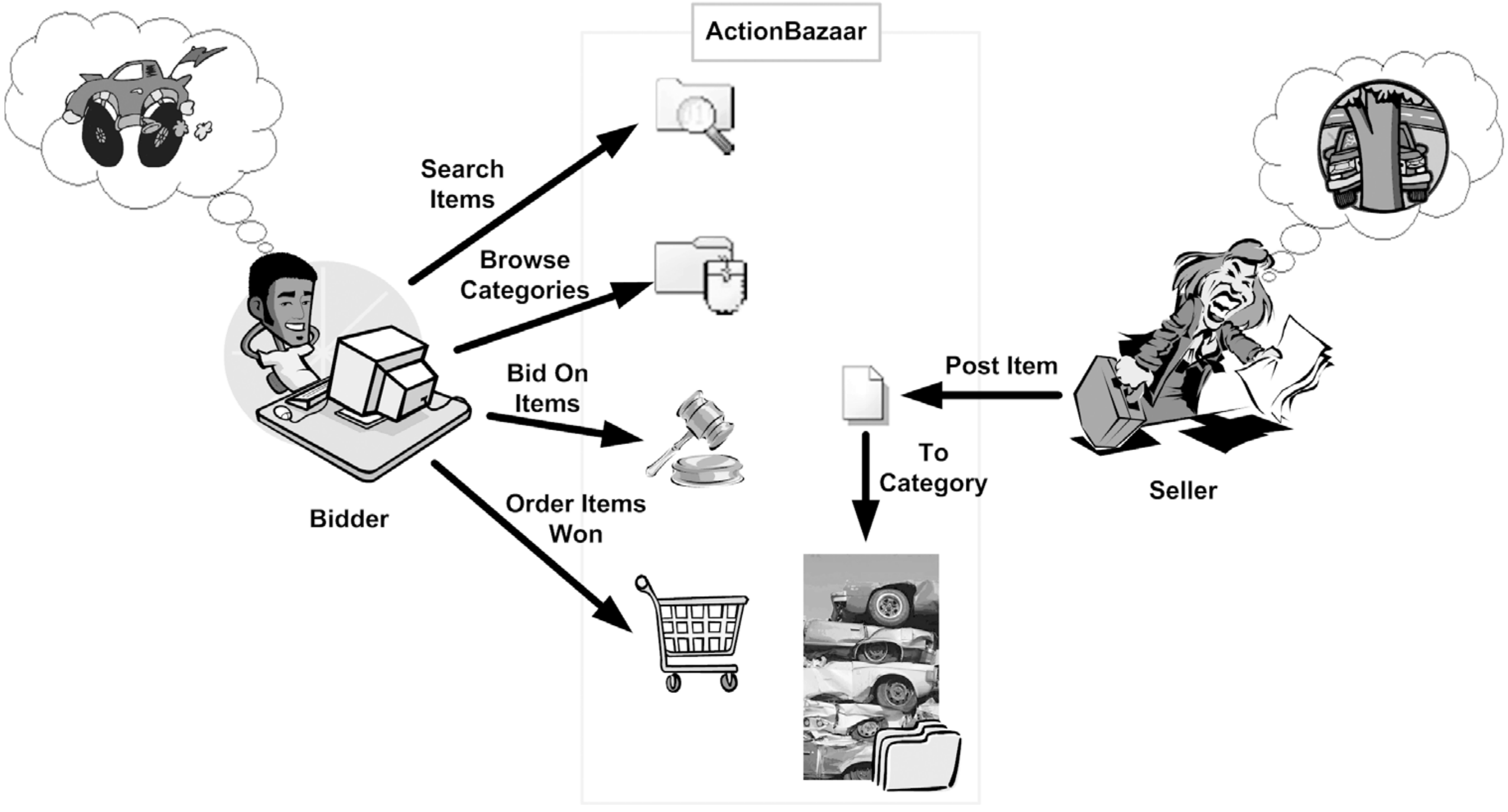
pictures: sxc.hu, pixabay

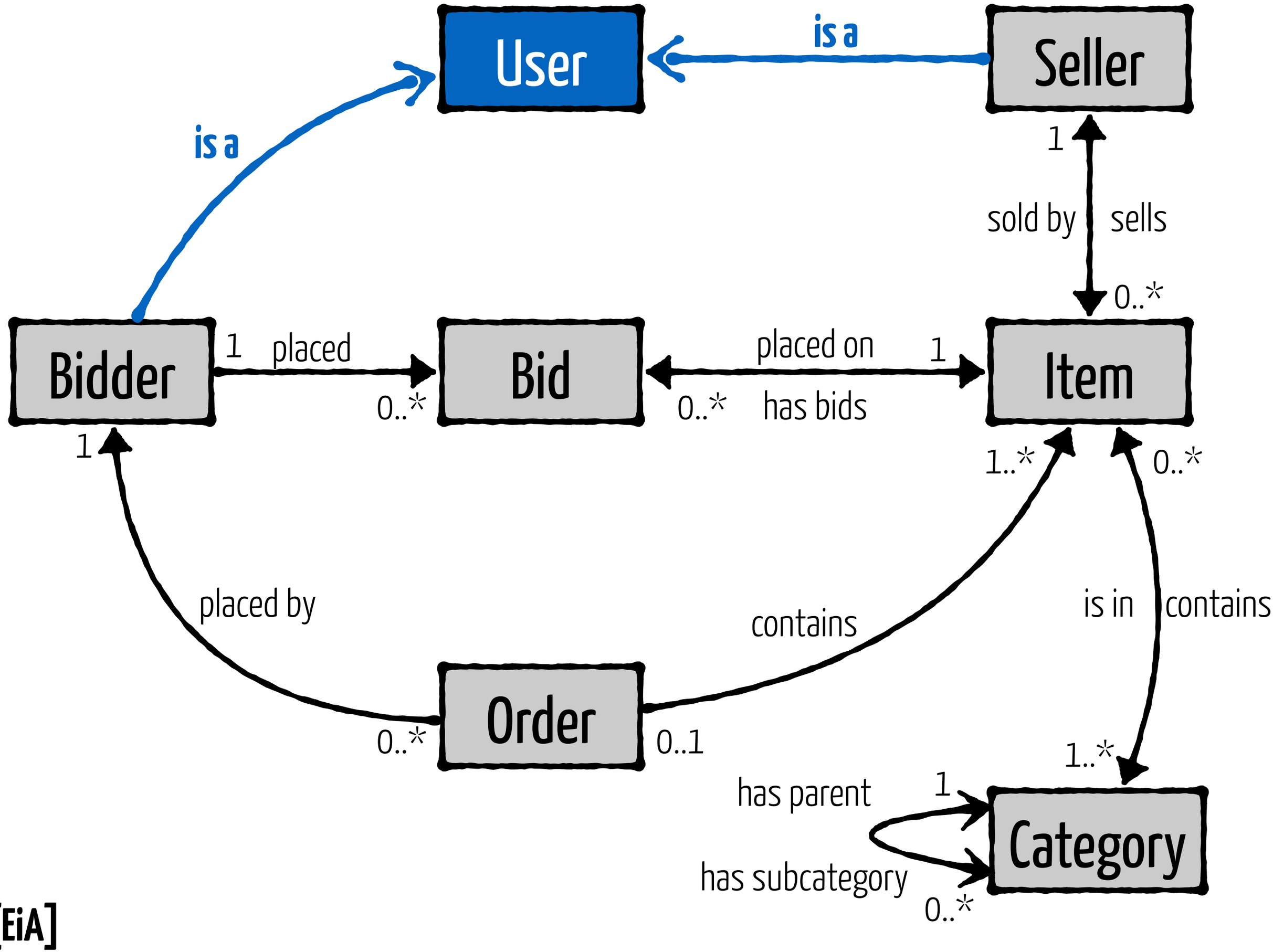
Object-Relational Mapping

Principles & Patterns

Session & Entity




ActionBazaar
Search Items
Browse Categories
Bid On Items
Order Items Won
Bidder
Post Item
To Category
Seller


User
is a
Seller
is a
1
sold by
sells
0..*
Bidder
1
placed
0..*
Bid
placed on
1
0..*
has bids
Item
1
1..*
0..*
placed by
contains
is in
contains
Order
0..*
0..1
1..*
has parent
1
Category
has subcategory
0..*

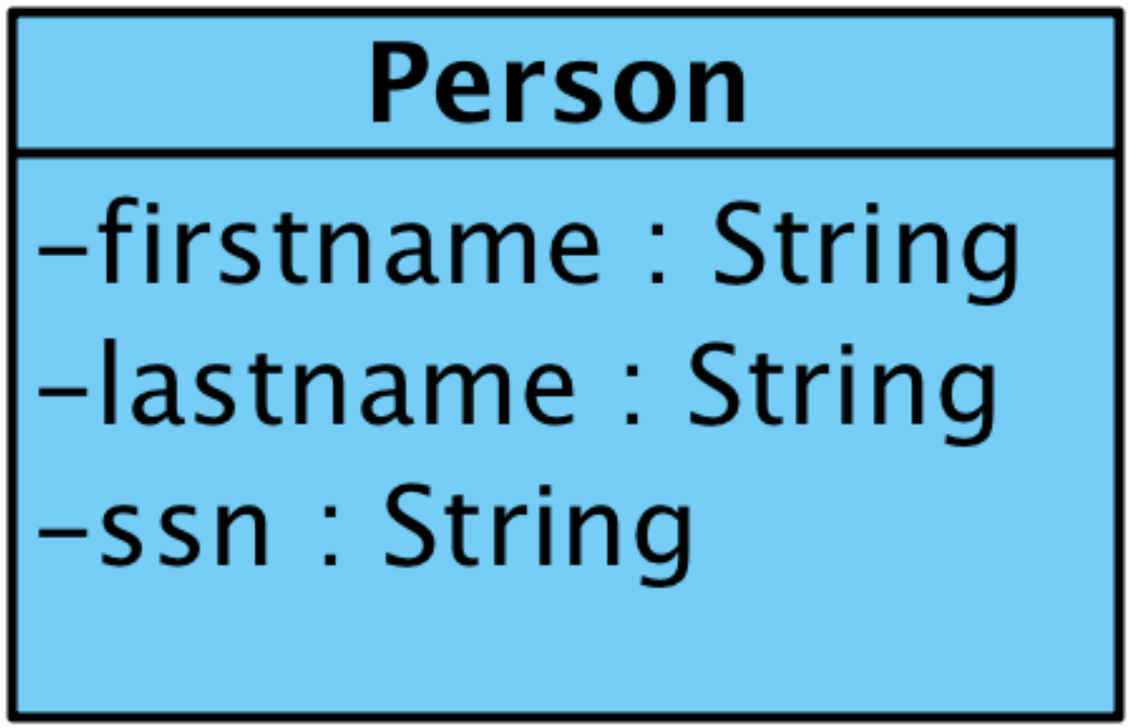
[EiA]

Impedance Mismatch

Object-Oriented	Relational
Classes	Relation (table)
Object	Tuple (row)
Attribute	Attribute (column)
Identity	Primary Key
Reference	Foreign Key
Inheritance	N/A
Methods	~ Stored Procedure

[EJB3 in Action]

Example of **Domain Model**



first_name	last_name	ssn
Sébastien	MOSSER	16118325358
...		

[PoEAA]

EJB Entities need more than simple annotations:

- An empty constructor
- A proper equals method that relies on business elements
- A proper hashCode method to support objects identification in caches

Category

```
@Entity
public class Category {

   public Category() { … }

   protected String name;

   public String getName() {
      return this.name;
   }

   public void setName(String n) {
      this.name = n.toUpperCase();
   }
}
```

property-based access (JPA)

Categor

```
@Entity
public class Category {

   public Category() { … }

   public String name;
}
```

Dot not use public fields

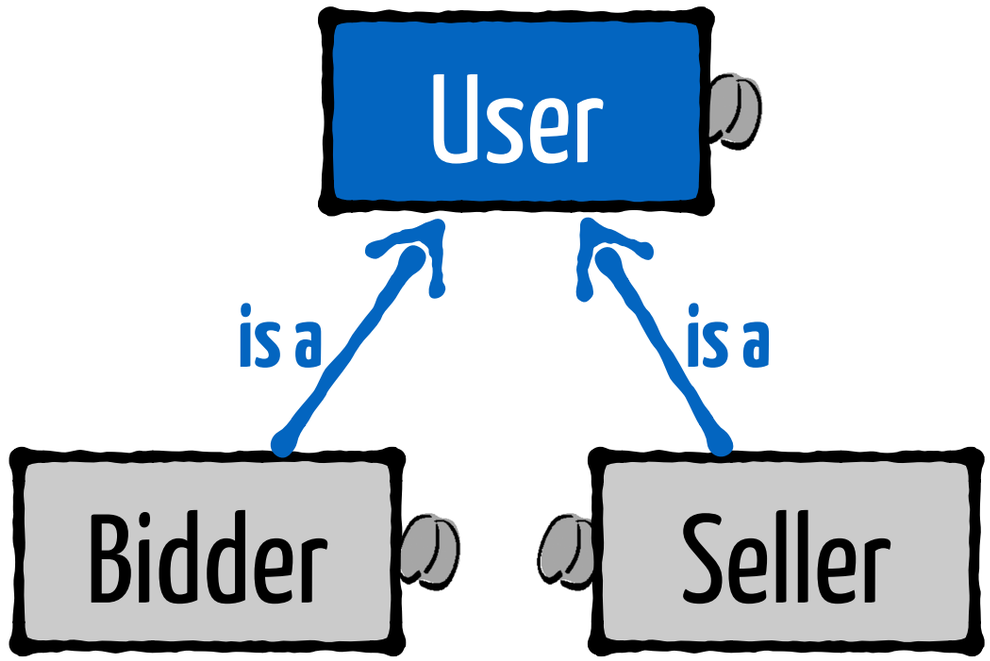
Annotate private fields or getters

But annotate them

with only 1 approach

TCF is using annotations on fields

```java
@Entity
public abstract class User {
   // …
}
```




```java
@Entity
public class Bidder extends User {
   // …
}

@Entity
public class Seller extends User {
   // …
}
```

Simple Primary Key: @Id

```
@Entity
public class Category {
   // …

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   public Long id;

}
```

Identifiers must define an "equals" method

Composite Key: @IdClass

```
public class CategoryPK extends Serializable {
   String name;
   Date createDate;
}
```

```
@Entity
@IdClass(CategoryPK.class)
public class Category {

   @Id
   protected String name;

   @Id
   protected Date createDate;
}
```

Identifiers must define an "equals" method

```
public class CategoryPK extends Serializable {

   public boolean equals(Object other) {

      if (other instanceOf CategoryPK) {
         final CategoryPK that = (CategoryPK) other;
         return that.name.equals(name) &&
                that.createDate.equals(createDate);
      return false;

   }

   public int hashCode() {
      return super.hashCode();
   }
}
```

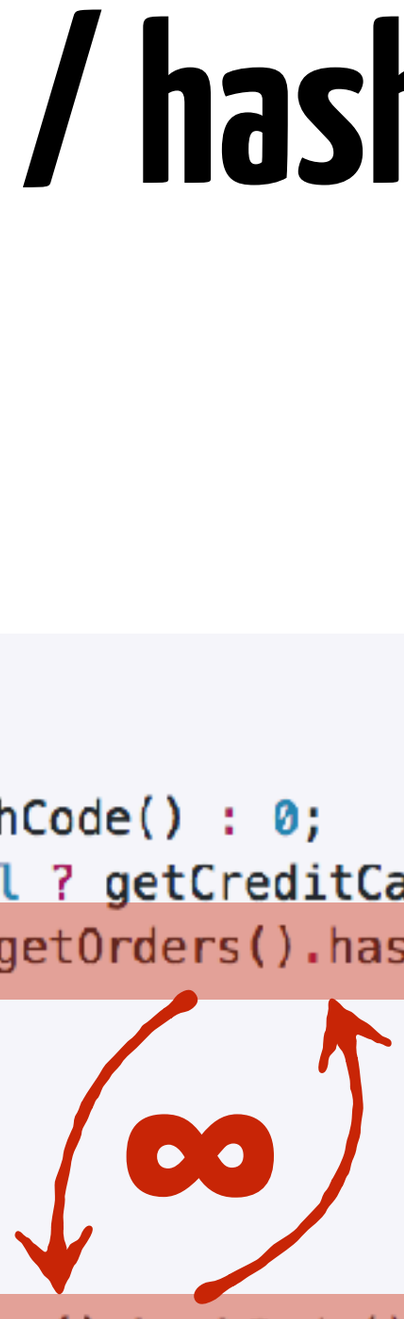
[EiA]

Serializable requirements

Auto-generated equals / hashcode

```
// Customer
public int hashCode() {
        int result = getName() != null ? getName().hashCode() : 0;
        result = 31 * result + (getCreditCard() != null ? getCreditCard().hashCode() : 0);
        result = 31 * result + (getOrders() != null ? getOrders().hashCode() : 0);
        return result;
}

// Order
public int hashCode() {
        int result = getCustomer() != null ? getCustomer().hashCode() : 0;
        result = 31 * result + (getItems() != null ? getItems().hashCode() : 0);
        result = 31 * result + (getStatus() != null ? getStatus().hashCode() : 0);
        return result;
}
```



Never ever use a database primary key as part of your business object equality definition

Equals is used when:

- putting objects in Sets
- when reattaching entities to a new persistence context

Embeddable Objects

```
@Embeddable
public class Address {
  protected String street;
  protected String city;
  protected String zipcode;
}

@Entity
public class User {

  @Id
  protected Long id;

  @Embedded
  protected Address address;
}
```

does not need an UID

Shared Identifier