*Advanced Java Programming

Java Persistence API part.2

Eriq Muhammad Adams J
eriq.adams@ub.ac.id | http://eriq.lecture.ub.ac.id

Agenda

* JPQL (Java Persistence Query Language)
* DAO Pattern
* Façade Pattern

JPQL (Java Persistence Query Language)

- Dynamic Query with JPQL
  - entityManager.createQuery(”select object o from Inventory o”).getResultList();
  - Binding parameter :
    - entityManager.createQuery(”select object o from Inventory o where o.year =:year”).setParameter(”year”, year).getResultList();
    - entityManager.createQuery(”select object o from Inventory o where o.year =?1”).setParameter(0, year).getResultList();

JPQL (Java Persistence Query Language) (cont.)

* Named Queries

```
@Entity
@NamedQueries({ @NamedQuery(name="findAllInventory",
queryString="select object(o) from Inventory o"),
@NamedQuery(name="findInventoryByYear", queryString="select
object(o) from Inventory o where o.year=:year"),
@NamedQuery(name="findInventoryByRegion", queryString="select
object(o) from Inventory o where o.region=?1 ")})
public class Inventory implements Serializable{
.....
```

JPQL (Java Persistence Query Language) (cont.)

- entityManager.createNamedQuery(”findAllInventory”).getResultList();
- entityManager.createNamedQuery(”findInventoryByYear”).setParameter(” year”, year).getResultList();
- entityManager.createNamedQuery(”findInventoryByRegion”).setParameter(0, region).getResultList();

JPQL (Java Persistence Query Language) (cont.)

* Bulk Update and Delete Operation

```
public int bulkDeleteEmptyInventory() {
    return em.createQuery("delete from Inventory o
where o.quantity = 0").executeUpdate();
}
```

JPQL (Java Persistence Query Language) (cont.)

- Native SQL Query
  - entityManager.createNativeQuery(”select * from inventory”, entity.Inventory.class).getResultList();
  - entityManager.createNativeQuery(”select * from inventory where year=?”, entity.Inventory.class).setParameter(0, year).getResultList();
  - **Resultset Mapping**
    - @SqlResultSetMapping(name = “InventoryResults", entities = @EntityResult(entityClass = Entity.Inventory.class));
    - entityManager.createNativeQuery(”select * from inventory”, InventoryResults).getResultList();

JPQL (Java Persistence Query Language) (cont.)

* Native Named SQL Query

```
@NamedNativeQuery(
name = "findInventoryByYear",
query = "SELECT *
  FROM inventory
  WHERE year = ?)",
resultClass = entity.Inventory.class)
@NamedNativeQuery(
name = "findInventoryByYear",
query = "SELECT *
  FROM inventory
  WHERE year = ?)",
resultSetMapping = "InventoryResults")
```

- entityManager.createNamedQuery(”findInventoryByYear”).setParamater(0,year).getResultList();

DAO Pattern

* Stand for Data Access Object Pattern
* Separate or decouple data access with any business logic code.
* It’s useful and widely used pattern because its reduce maintenance problem.

DAO Pattern (cont.)

* Illustration Sample

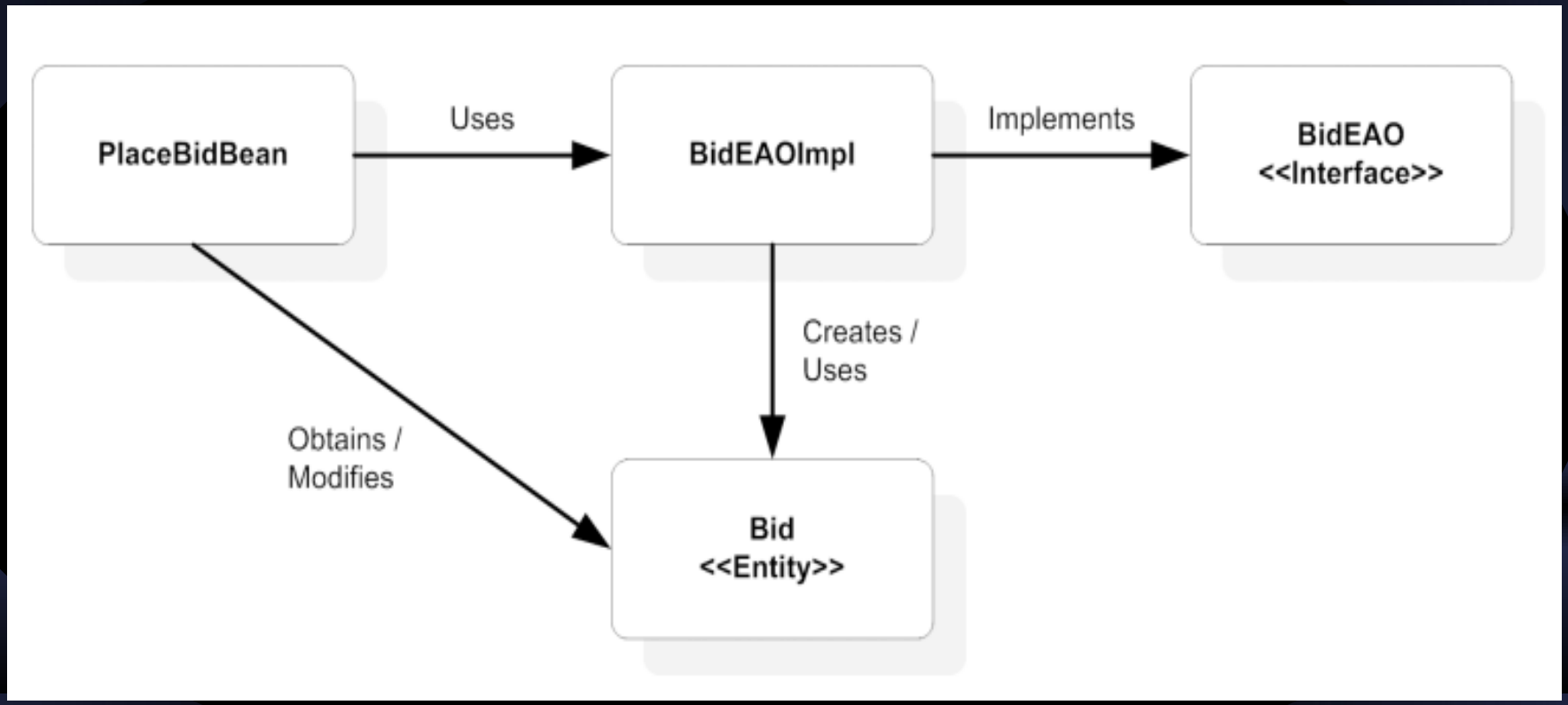

Fig. taken from EJB 3 in Action, Manning

Façade Pattern

* The primary reasons the Session Façade pattern was invented was to reduce the number of remote calls for previous EJB incarnations, and this still holds true for EJB 3.

Façade Pattern (cont.)

* Illustration sample

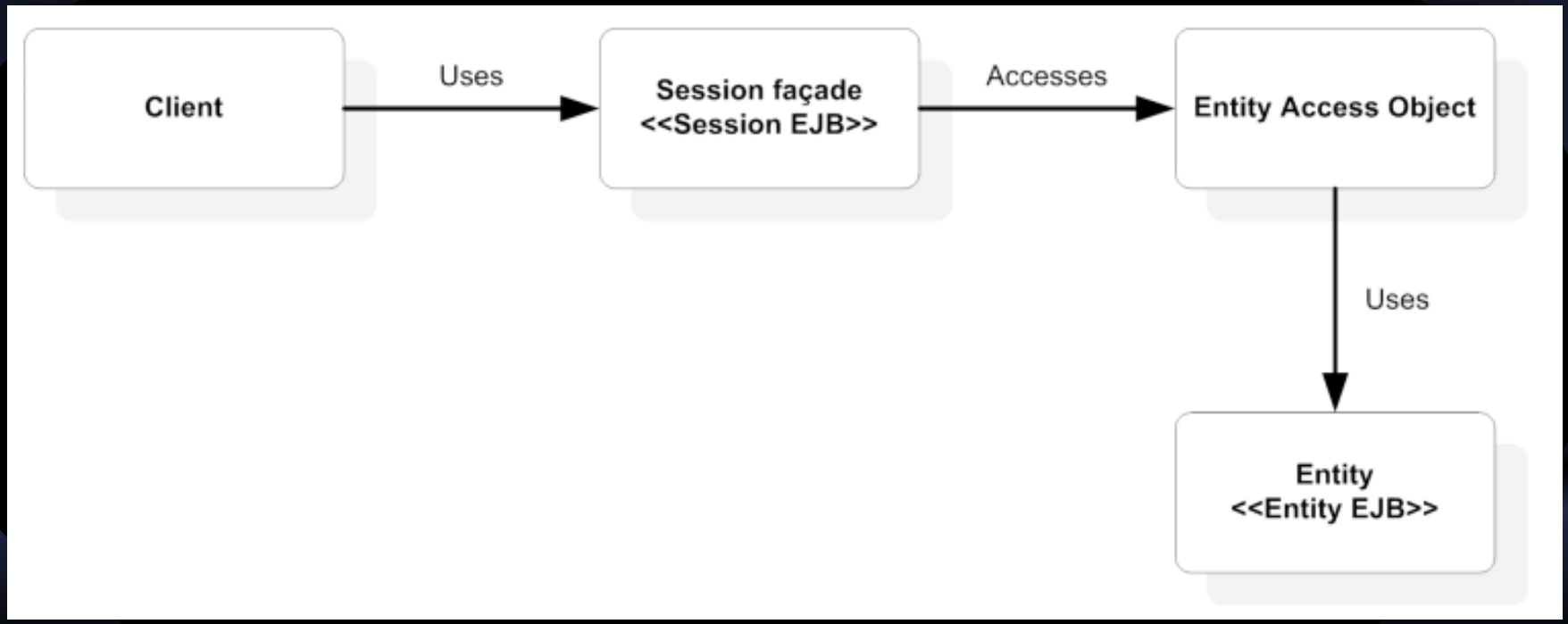

Fig. taken from EJB 3 in Action, Manning

References

* EJB 3 in Action, Manning
* Beginning EJB 3 Application Development, Apress