Hibernate Cheat Sheet

ORM · Session · HQL · Caching · Associations

worksheetdownload.com

SETUP & CONFIG

```
<dep>hibernate-core 6.x</dep>
hibernate.cfg.xml OR persistence.xml
hibernate.dialect=PostgreSQLDialect
hibernate.connection.url=jdbc:postgresql://...
hibernate.hbm2ddl.auto=validate (prod)
```

SESSION FACTORY / SESSION

```
SessionFactory sf = new Configuration()
.configure().buildSessionFactory();
try(Session s = sf.openSession()){
s.beginTransaction();
s.persist(entity);
s.getTransaction().commit();
}
```

ENTITY MAPPING

```
@Entity @Table(name="users")
public class User {
@Id @GeneratedValue(strategy=IDENTITY)
private Long id;
@Column(nullable=false, unique=true)
private String email;
}
```

ASSOCIATIONS

```
@OneToMany(mappedBy="user", cascade=ALL)
private List<Order> orders = new
ArrayList<>();
@ManyToOne @JoinColumn(name="user_id")
private User user;
@ManyToMany @JoinTable(...)
```

Owning side has @JoinColumn / @JoinTable

HQL QUERIES

```
Query<User> q = session.createQuery(
"FROM User WHERE email = :email",
User.class);
q.setParameter("email", email);
List<User> users = q.list();
session.get(User.class, id)  // load by
PK
```

HQL uses entity names, not table names

CRITERIA API

```
CriteriaBuilder cb =
session.getCriteriaBuilder();
CriteriaQuery<User> cq =
cb.createQuery(User.class);
Root<User> r = cq.from(User.class);
cq.select(r).where(cb.equal(r.get("active"),true));
session.createQuery(cq).list();
```

Type-safe, refactor-friendly alternative to HQL

LAZY vs EAGER LOADING

LAZY (default for collections): loads on access

EAGER: loads with parent query (N+1 risk)

```
@OneToMany(fetch = FetchType.LAZY)
```

Fix N+1: JOIN FETCH in HQL

```
"FROM User u JOIN FETCH u.orders"
@BatchSize(size=20) for batch loading
```

CACHING

L1 Cache: per-session, automatic

L2 Cache: shared across sessions

```
@Cacheable  (entity or collection)
hibernate.cache.region.factory_class=EhcacheRegionFactory
@Cache(usage=READ_WRITE)
```

Query cache: .setCacheable(true) on query

TRANSACTIONS

```
session.beginTransaction();
// ... do work
session.getTransaction().commit();
session.getTransaction().rollback(); //
on error
```

Use JTA for distributed transactions

Never use AUTO flush mode in production

CASCADE & ORPHAN REMOVAL

```
cascade = {CascadeType.PERSIST, MERGE,
REMOVE}
cascade = CascadeType.ALL  // use with
care
orphanRemoval = true  // deletes removed
children
@OneToMany(cascade=ALL,
orphanRemoval=true)
```

Only set on owning side; avoid cascade from child to parent

COMMON ANNOTATIONS

```
@Entity @Table @Column @Id
@GeneratedValue
@OneToOne @OneToMany @ManyToOne
@ManyToMany
@JoinColumn @JoinTable @Embedded
@Embeddable
@Temporal(DATE/TIME/TIMESTAMP)
@Enumerated(STRING/ORDINAL)
@Version  (optimistic locking)
```

COMMON GOTCHAS

LazyInitializationException session closed before access

N+1 problem with EAGER or no JOIN FETCH

cascade=ALL + orphanRemoval on ManyToMany = data loss

Detached entity passed to persist = error

hbm2ddl.auto=create drops tables on startup!