CS544 EA

Hibernate

EntityManager: Cache

Insert may be Held in Cache

- With .persist()
  - Hibernate pushes to the DB right away for @GeneratedValue entities
  - Hibernate holds it in cache until tx.commit() for assigned IDs

```
@Entity
public class Person {
	@Id
	@GeneratedValue
	private Long id;
	private String name;
```

Generated ID

```
em.getTransaction().begin();
Person p = new Person("Aaron James");
System.out.println("1");
em.persist(p);
System.out.println("2");
em.getTransaction().commit();
```

```
1
Hibernate: insert into Person (name) values (?)
2
```

```
@Entity
public class Person {
	@Id
	private Long id;
	private String name;
```

```
em.getTransaction().begin();
Person p = new Person("Aaron James");
p.setId(1L);
System.out.println("1");
em.persist(p);
System.out.println("2");
em.getTransaction().commit();
```

Assigned ID

Held in cache until .commit()

```
1
2
Hibernate: insert into Person (name, id) values (?, ?)
```

Retrievals use cache

- .find() and .getReference() **do not hit** the DB
  - If the object is **already in cache**

```
@Entity
public class Person {
	@Id
	@GeneratedValue
	private Long id;
	private String name;
```

```
em.getTransaction().begin();
Person p = new Person("Aaron James");
System.out.println("1");
em.persist(p);
System.out.println("2");
long id = p.getId();
System.out.println("3");
em.find(Person.class, id);
System.out.println("4");
em.getReference(Person.class, id);
System.out.println("5");
em.getTransaction().commit();
System.out.println("6");
```

```
1
Hibernate: insert into Person (name) values (?)
2
3
4
5
6
```

Updates are held in cache

- Updates to managed objects are **pushed** on transaction **commit**

```
@Entity
public class Person {
	@Id
	@GeneratedValue
	private Long id;
	private String name;
```

Generated ID

```
em.getTransaction().begin();
Person p = new Person("Aaron James");
System.out.println("1");
em.persist(p);
System.out.println("2");
p.setName("Updated Name");
System.out.println("3");
em.getTransaction().commit();
System.out.println("4");
```

Update held until .commit()

```
1
Hibernate: insert into Person (name) values (?)
2
3
Hibernate: update Person set name=? where id=?
4
```

Removals ‘held’ in cache

- Removed objects are **marked for deletion**
  - No longer officially held in cache
  - But DELETE not executed until tx.commit()

.contains() returns false

```java
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();
Person p = new Person("Aaron James");
System.out.println("1");
em.persist(p);
System.out.println("2");
em.remove(p);
System.out.println("3");
em.getTransaction().commit();
System.out.println("4");
```

Remove held until .commit()

```
1
Hibernate: insert into Person (name) values (?)
2
3
Hibernate: delete from Person where id=?
4
```

Changes Pushed Before Query

- All changes in cache are **pushed before** executing a **query**

```
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();
Person p = new Person("Aaron James");
System.out.println("1");
em.persist(p);
System.out.println("2");
p.setName("Updated Name");
System.out.println("3");
em.remove(p);
System.out.println("4");
TypedQuery<Person> q = em.createQuery("from Person", Person.class);
System.out.println("5");
List<Person> people = q.getResultList();
System.out.println("6");
em.getTransaction().commit();
System.out.println("7");
```

This behavior can be changed by setting the **FlushMode**

Changes can be:
inserts, updates, deletes
held in cache

Update not done
because entity removed

```
1
Hibernate: insert into Person (name) values (?)
2
3
4
5
Hibernate: delete from Person where id=?
Hibernate: select person0_.id as id1_0_,
person0_.name as name2_0_ from Person person0_
6
7
```

.flush()

- You can **tell** the entity manager to **flush** changes
  - Instead of waiting for .commit() or a query

```
em.getTransaction().begin();
Person p = new Person("Aaron James");
System.out.println("1");
em.persist(p);
System.out.println("2");
em.remove(p);
System.out.println("3");
em.flush();
System.out.println("4");
TypedQuery<Person> q = em.createQuery("from Person", Person.class);
System.out.println("5");
List<Person> people = q.getResultList();
System.out.println("6");
em.getTransaction().commit();
System.out.println("7");
```

Changes can be:
inserts, updates, deletes
held in cache

```
1
Hibernate: insert into Person (name) values (?)
2
3
Hibernate: delete from Person where id=?
4
5
Hibernate: select person0_.id as id1_0_,
person0_.name as name2_0_ from Person person0_
6
7
```

.refresh()

- .refresh() ‘refreshes’ the data in the entity with the **values found in the DB**
  - Data in the DB may have changed
  - Can be used to undo updates

Usually not because EntityManager lifetime should be short

```
em.getTransaction().begin();
Person p = new Person("Aaron James");
System.out.println("1");
em.persist(p);
System.out.println("2");
Thread.sleep(5000); // sleep for 5 secs (other program changes db)
System.out.println("3");
// tries to 'get again' from db, but receives cached version
p = em.find(Person.class, p.getId());
System.out.println(p.getName());
System.out.println("4");
em.refresh(p); // forced to go to db again
System.out.println(p.getName());
em.getTransaction().commit();
```

```
1
Hibernate: insert into Person (name) values (?)
2
3
Aaron James
4
Hibernate: select person0_.id as id1_0_0_,
person0_.name as name2_0_0_ from Person
person0_ where person0_.id=?
Updated Name
```

.contains()

- .contains() **checks** if the object is in the **cache**
  - Both assigned and generated are in cache right away
  - Assigned not in DB until commit

```java
@Entity
public class Person {
	@Id
	@GeneratedValue
	private Long id;
	private String name;
```

Generated ID

```java
em.getTransaction().begin();
Person p = new Person("Aaron James");
System.out.println(em.contains(p));
em.persist(p);
System.out.println(em.contains(p));
em.getTransaction().commit();
```

```
false
Hibernate: insert into Person (name) values (?)
true
```

```java
@Entity
public class Person {
	@Id
	private Long id;
	private String name;
```

Assigned ID

```java
em.getTransaction().begin();
Person p = new Person("Aaron James");
p.setId(1L);
System.out.println(em.contains(p));
em.persist(p);
System.out.println(em.contains(p));
em.getTransaction().commit();
```

```
false
true
Hibernate: insert into Person (name, id) values (?, ?)
```

.detach()

- .detach() detaches **an entity** from the cache
  - Entity state is then detached
  - .contain() no longer finds it

```
em.getTransaction().begin();
Person p1 = new Person("John");
Person p2 = new Person("Jane");
em.persist(p1);
em.persist(p2);
em.detach(p1);
System.out.println(em.contains(p1));
System.out.println(em.contains(p2));
em.getTransaction().commit();
```

```
Hibernate: insert into Person (name) values (?)
Hibernate: insert into Person (name) values (?)
false
true
```

.clear()

- .clear() removes **all entities** from the cache
  - All entity objects are detached
  - The cache is empty

```
em.getTransaction().begin();
Person p1 = new Person("John");
Person p2 = new Person("Jane");
em.persist(p1);
em.persist(p2);
em.clear();
System.out.println(em.contains(p1));
System.out.println(em.contains(p2));
em.getTransaction().commit();
```

```
Hibernate: insert into Person (name) values (?)
Hibernate: insert into Person (name) values (?)
false
false
```

.close()

- .close() closes the EntityManager
  - All entities are **automatically detached**
  - Can no longer use the EntityManager