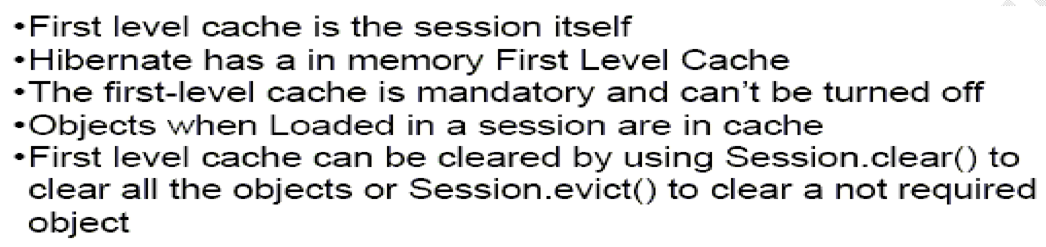
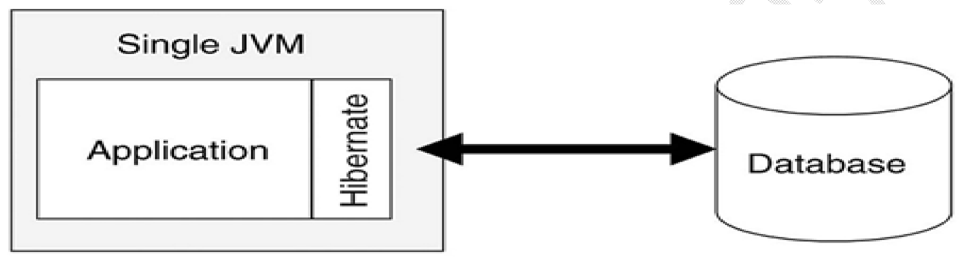
Cache:

Caching is nothing but some buffer where a record is stored when first time retrieved from the database.
When second time needed the same record, Hibernate does not access the database and instead reads from the cache.

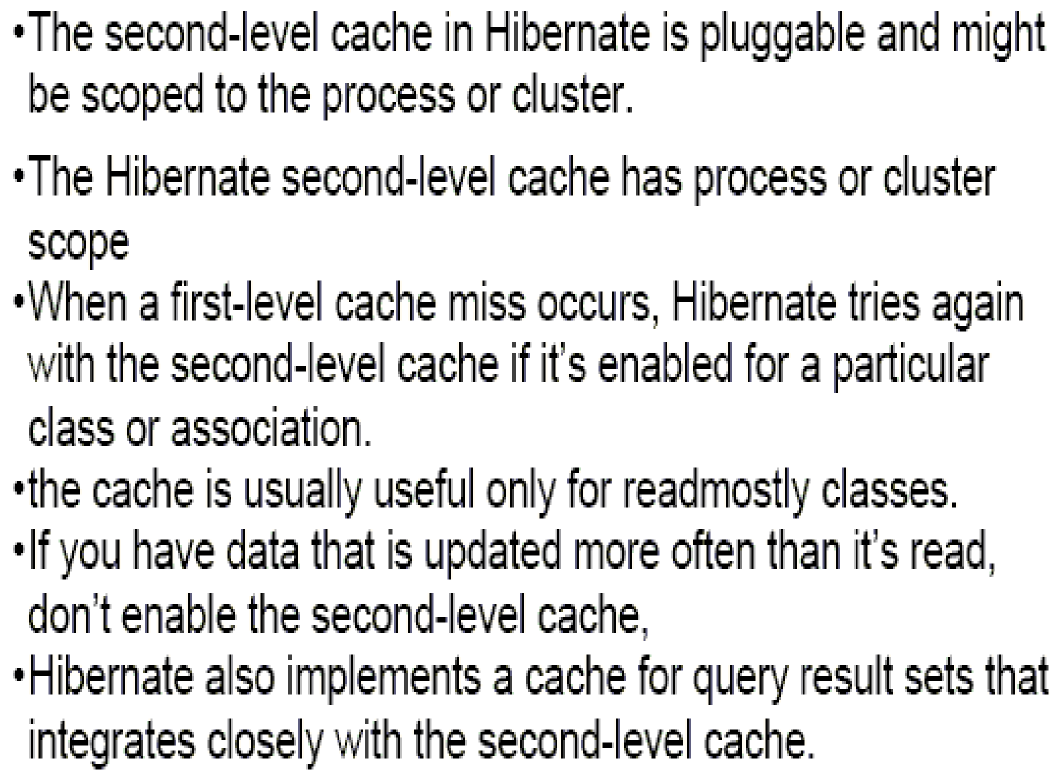
Hibernate First Level Cache



- First level cache is the session itself
- Hibernate has a in memory First Level Cache
- The first-level cache is mandatory and can't be turned off
- Objects when Loaded in a session are in cache
- First level cache can be cleared by using Session.clear() to clear all the objects or Session.evict() to clear a not required object



Hibernate Second Level Cache:

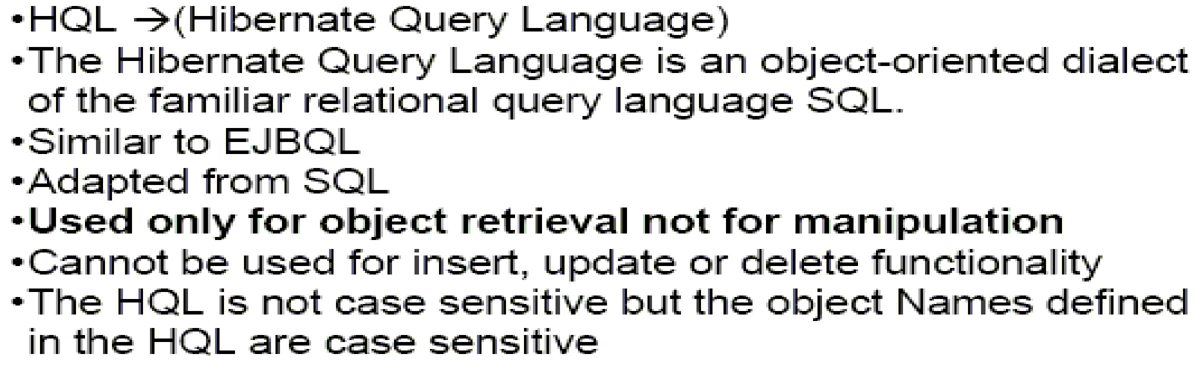


- The second-level cache in Hibernate is pluggable and might be scoped to the process or cluster.
- The Hibernate second-level cache has process or cluster scope
- When a first-level cache miss occurs, Hibernate tries again with the second-level cache if it's enabled for a particular class or association.
- the cache is usually useful only for readmostly classes.
- If you have data that is updated more often than it's read, don't enable the second-level cache,
- Hibernate also implements a cache for query result sets that integrates closely with the second-level cache.

Hibernate comes with four open-source cache implementations to support second-level caching.

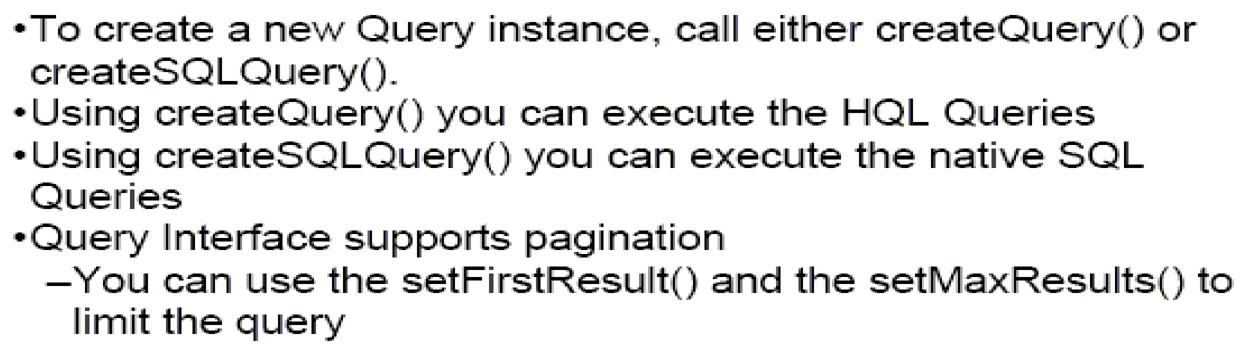
1. **EHCache (Easy Hibernate Cache)**
   2. OSCache (Open Symphony Cache)
   3. Swarm Cache
   4. JBoss Tree Cache.

Introduction to HQL:



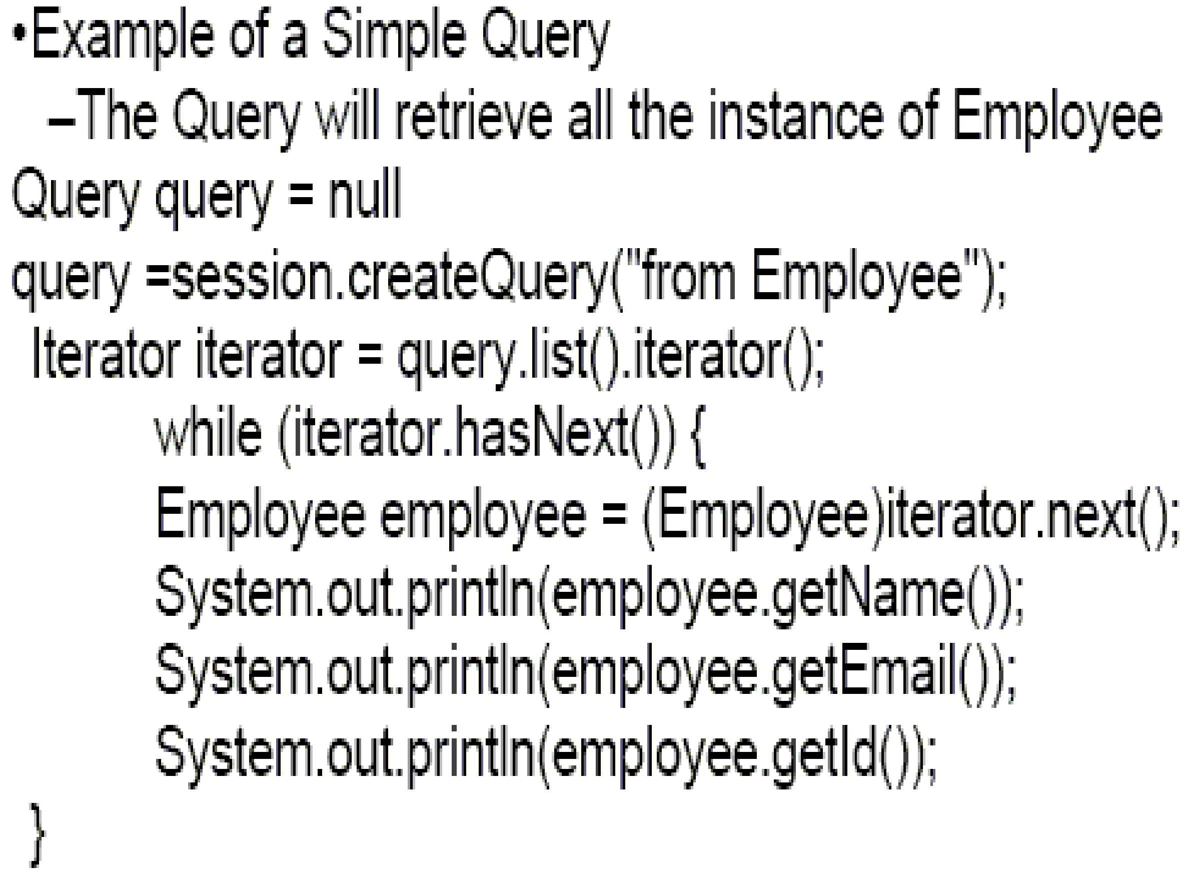
- HQL →(Hibernate Query Language)
- The Hibernate Query Language is an object-oriented dialect of the familiar relational query language SQL.
- Similar to EJBQL
- Adapted from SQL
- **Used only for object retrieval not for manipulation**
- Cannot be used for insert, update or delete functionality
- The HQL is not case sensitive but the object Names defined in the HQL are case sensitive

Query Interface:



- To create a new Query instance, call either createQuery() or createSQLQuery().
- Using createQuery() you can execute the HQL Queries
- Using createSQLQuery() you can execute the native SQL Queries
- Query Interface supports pagination
  - –You can use the setFirstResult() and the setMaxResults() to limit the query

An example of simple Query:



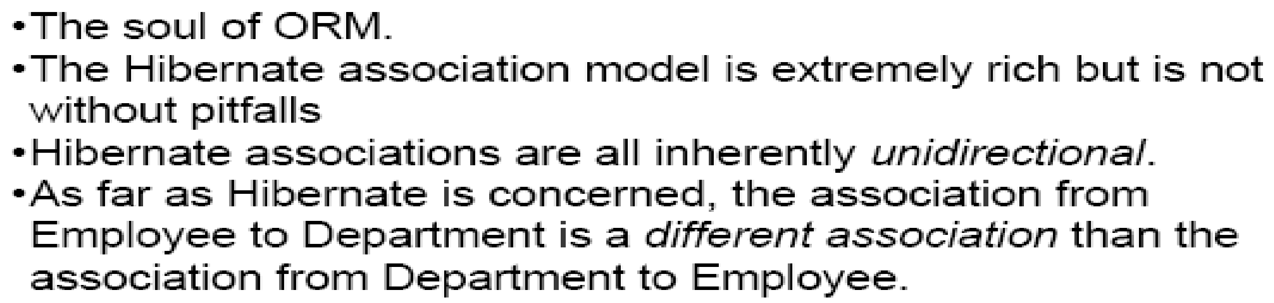
- Example of a Simple Query
  - –The Query will retrieve all the instance of Employee

```
Query query = null
query =session.createQuery("from Employee");
 Iterator iterator = query.list().iterator();
        while (iterator.hasNext()) {
        Employee employee = (Employee)iterator.next();
        System.out.println(employee.getName());
        System.out.println(employee.getEmail());
        System.out.println(employee.getId());
 }
```

- **HQL supports:**
- **WHERE clause**
- **ORDER BY clause**
- **GROUP BY clause**
- **All types of joins (inner, left outer, right outer, outer)**

- **Subquery**

<u>Associations</u>



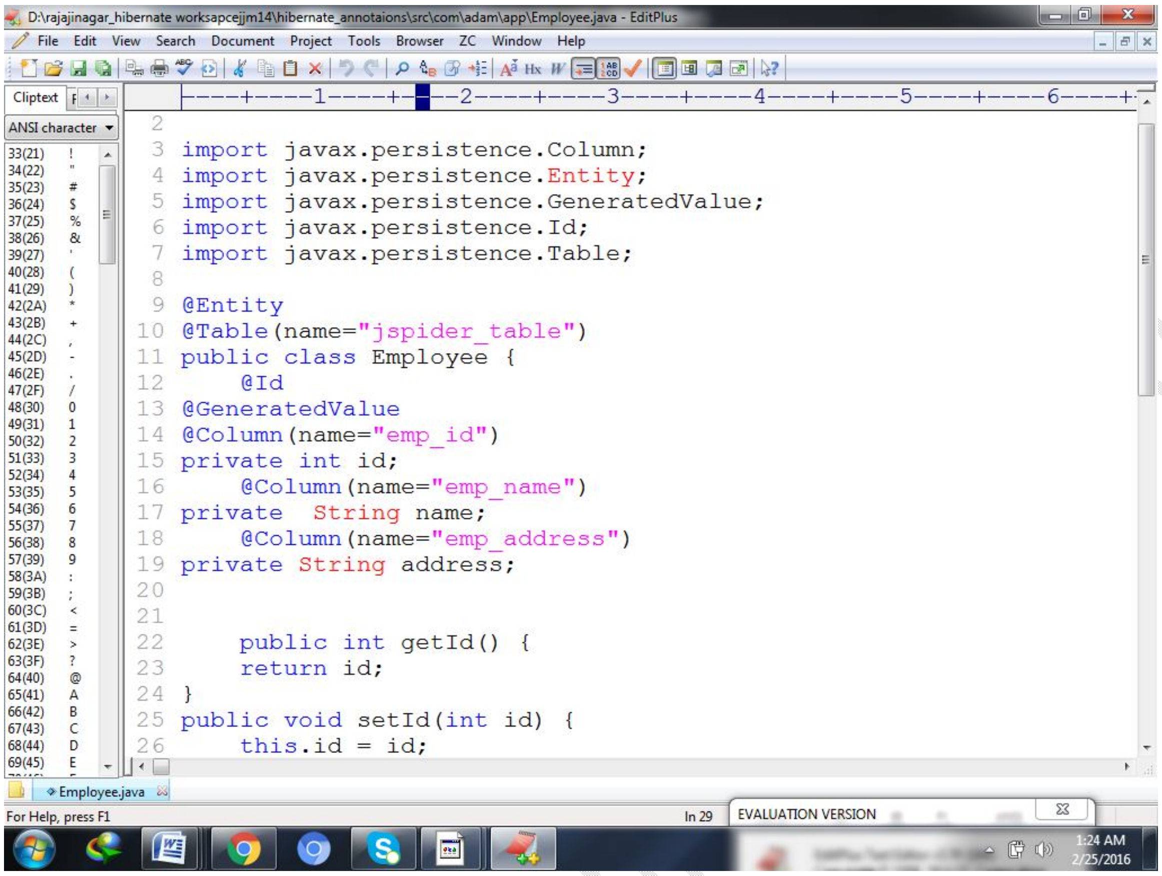
- The soul of ORM.
- The Hibernate association model is extremely rich but is not without pitfalls
- Hibernate associations are all inherently *unidirectional*.
- As far as Hibernate is concerned, the association from Employee to Department is a *different association* than the association from Department to Employee.

Types of Associations with Annotations

- ONE-TO-ONE -----→@OneToOne
- ONE-TO-MANY-----→@OneToMany
- MANY-TO-ONE-----→@ManyToOne
- MANY-TO-MANY-----→@ManyToMany

Common Annotations :
@Entity
@Table
@Id
@GeneratedValue
@Column

Example on Hibernate Annotations


D:\rajajinagar_hibernate worksapcejjm14\hibernate_annotaions\src\com\adam\app\Employee.java - EditPlus
File Edit View Search Document Project Tools Browser ZC Window Help
Employee.java
For Help, press F1
EVALUATION VERSION

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="jspider_table")
public class Employee {
    @Id
@GeneratedValue
@Column(name="emp_id")
private int id;
    @Column(name="emp_name")
private  String name;
    @Column(name="emp_address")
private String address;


    public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
```