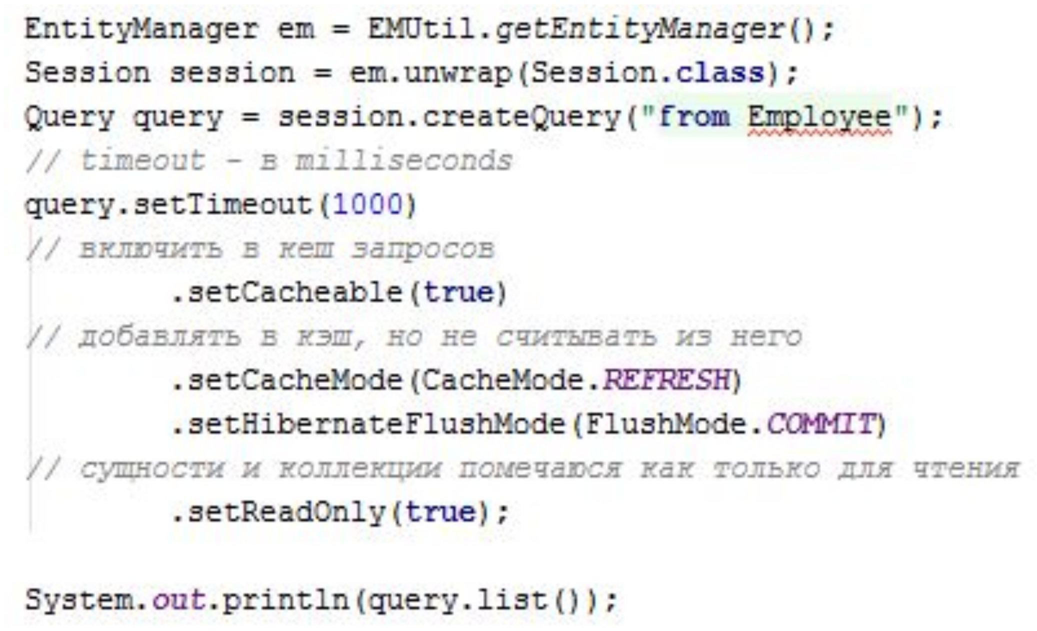
Hibernate
HQL / JPQL

Автор: Юлий Слабко

Hibernate Query Language (HQL)

- **Hibernate Query Language (HQL) -** это объектно ориентированный язык запросов, похожий на SQL, но вместо операций над таблицами и колонками, HQL работает с persistent objects и их свойствами.

FROM Clause



```
EntityManager em = EMUtil.getEntityManager();
Session session = em.unwrap(Session.class);
Query query = session.createQuery("from Employee");
// timeout - в milliseconds
query.setTimeout(1000)
// включить в кеш запросов
        .setCacheable(true)
// добавлять в кэш, но не считывать из него
        .setCacheMode(CacheMode.REFRESH)
        .setHibernateFlushMode(FlushMode.COMMIT)
// сущности и коллекции помечаюся как только для чтения
        .setReadOnly(true);

System.out.println(query.list());
```

FROM Clause

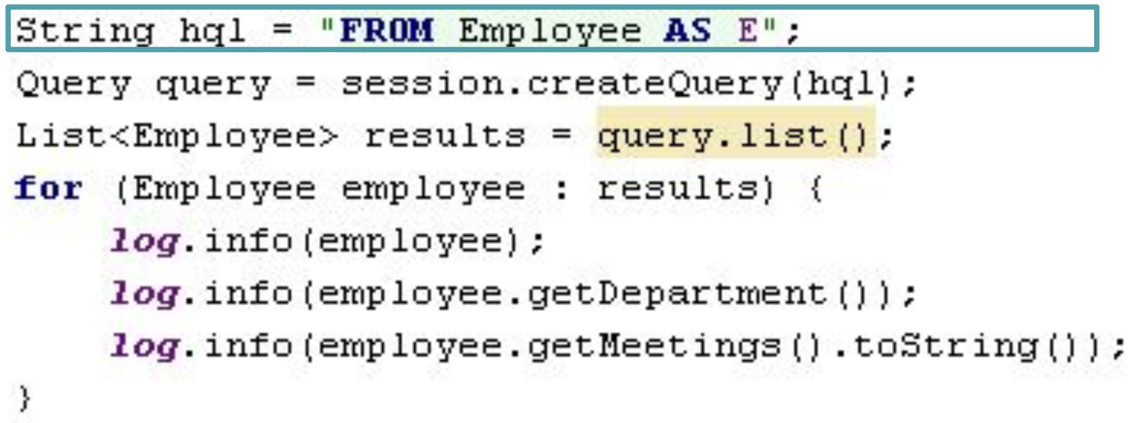
- Мы используем условие FROM, если мы хотим загрузить все объекты из базы данных в память.

```
@Test
public void selectTest() {
    EntityManager em = EMUtil.getEntityManager();
    Session session = em.unwrap(Session.class);
    Query query = session.createQuery("from Employee");
    query.list().forEach(System.out::println);
}
```

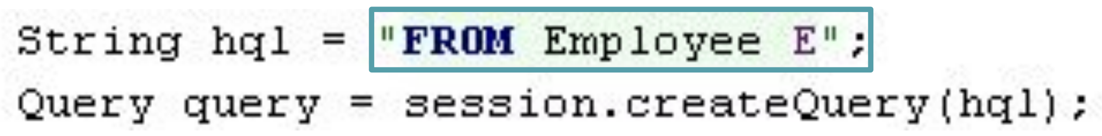
```
HQL -> select employee0_.id as id1_2_, employee0_.age as age2_2_, employee0_.name as name3_2_,
employee0_.salary as salary4_2_ from Employee employee0_
[Employee{id=1, name='Yulij, age=30, salary=8500},
Employee{id=2, name='Alex, age=28, salary=5500},
Employee{id=3, name='Sergey, age=40, salary=7500},
Employee{id=4, name='Yulij, age=40, salary=9500},
Employee{id=5, name='Maria, age=28, salary=3500}]
```

AS Clause

- Условие AS используется для алиасов классов в вашем HQL-запросе, особенно, если используются длинные запросы.



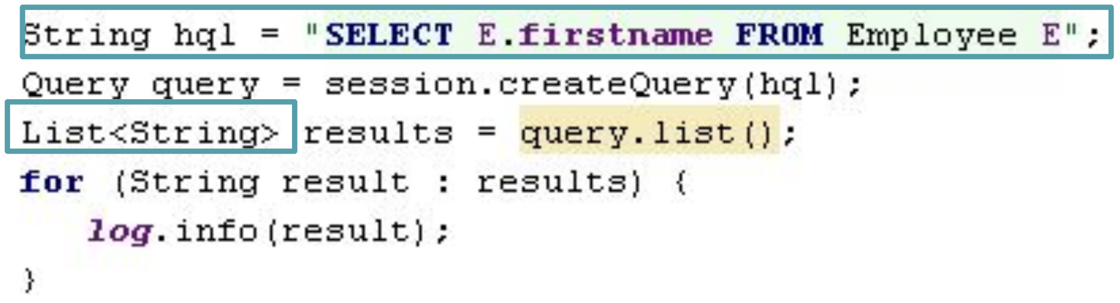
```
String hql = "FROM Employee AS E";
Query query = session.createQuery(hql);
List<Employee> results = query.list();
for (Employee employee : results) {
    log.info(employee);
    log.info(employee.getDepartment());
    log.info(employee.getMeetings().toString());
}
```



```
String hql = "FROM Employee E";
Query query = session.createQuery(hql);
```

SELECT Clause

- Условие Select предоставляет больше контроля над результатом вывода чем условие from. Если вы хотите вывести не все поля объекта, тогда используйте select.

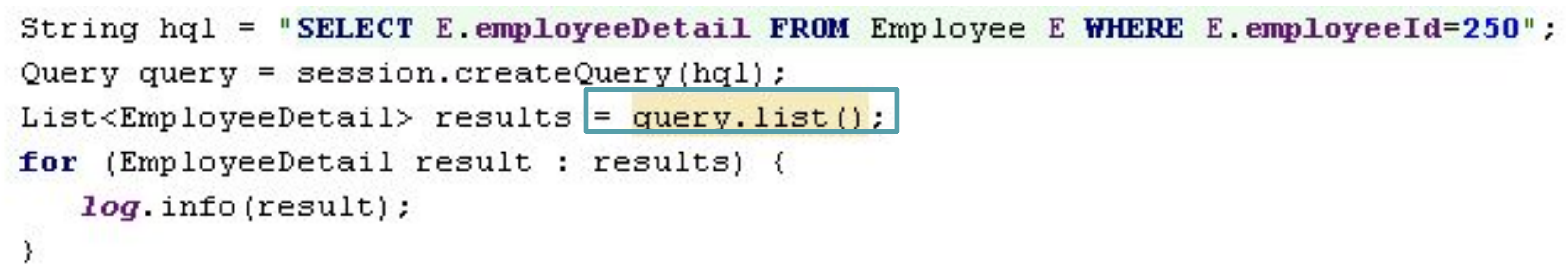


```
String hql = "SELECT E.firstname FROM Employee E";
Query query = session.createQuery(hql);
List<String> results = query.list();
for (String result : results) {
   log.info(result);
}
```

Hibernate: select employee0_.firstname as col_0_0_ from T_EMPLOYEE employee0_
2012-12-20 03:33:58,046 INFO - Yuli

SELECT Clause

- Вы можете доставать объекты внутри других объектов при помощи select.



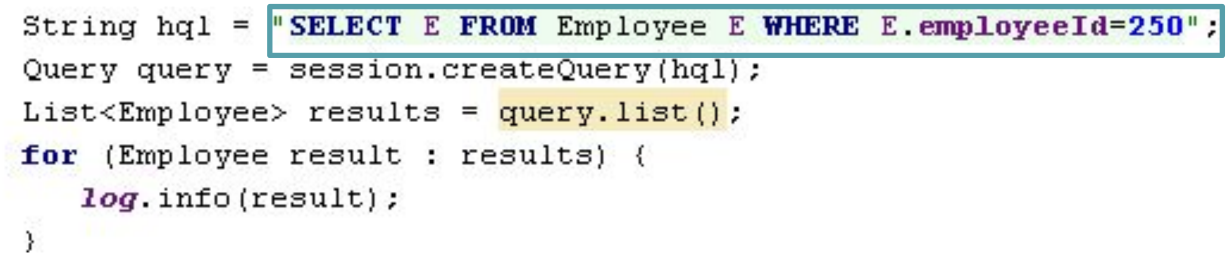
```
String hql = "SELECT E.employeeDetail FROM Employee E WHERE E.employeeId=250";
Query query = session.createQuery(hql);
List<EmployeeDetail> results = query.list();
for (EmployeeDetail result : results) {
    log.info(result);
}
```

select employeede1_.F_employeeId as F1_0_, employeede1_.city as city0_, employeede1_.country as country0_, employeede1_.state as state0_, employeede1_.street as street0_ from T_EMPLOYEE employee0_, T_EMPLOYEEDETAIL employeede1_ where employee0_.F_EMPLOYEE_ID=employeede1_.F_employeeId and employee0_.F_EMPLOYEE_ID=250

```
XX:XX:51,171 INFO  - EmployeeDetail{country='Belarus', employeeId=250,
street='Golodeda', city='Minsk', state='XXX'}
```

WHERE Clause

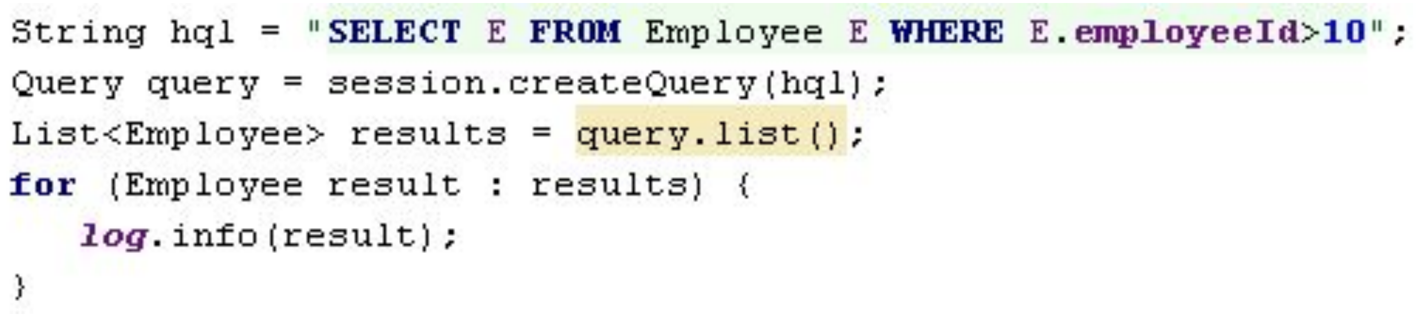
- Если вы хотите отфильтровать результат, то используйте условие where.



```
String hql = "SELECT E FROM Employee E WHERE E.employeeId=250";
Query query = session.createQuery(hql);
List<Employee> results = query.list();
for (Employee result : results) {
    log.info(result);
}
```

WHERE Clause

- Вы можете использовать ключевые слова после условия where:
- =, >=, <=, <>, !=, like
- in, not in, between, is null, is not null, is empty, is not empty, member of и not member of
- "Simple" case, case ... when ... then ... else ... end;
- and "searched" case,
- case when ... then ... else ... end
- current_date(), current_time(), and current_timestamp()
- substring(), trim(), lower(), upper(), abs(), sqrt(), bit_length(), mod()
- str() for converting numeric or temporal values to a readable string



```
String hql = "SELECT E FROM Employee E WHERE E.employeeId>10";
Query query = session.createQuery(hql);
List<Employee> results = query.list();
for (Employee result : results) {
    log.info(result);
}
```

ORDER BY Clause

- Для сортировки ваших результатов применяется условие Order BY с двумя параметрами:
- ASC – по возрастанию
- DESC – по убыванию

```
@Test
public void orderByTest() {
    EntityManager em = EMUtil.getEntityManager();
    Session session = em.unwrap(Session.class);
    Query query = session.createQuery("from Employee order by salary desc");
    query.list().forEach(System.out::println);
}
```

```
HQL -> select employee0_.id as id1_2_, employee0_.age as age2_2_, employee0_.name as name3_2_,
employee0_.salary as salary4_2_ from Employee employee0_ order by employee0_.salary desc
Employee{id=4, name='Yulij, age=40, salary=9500}
Employee{id=1, name='Yulij, age=30, salary=8500}
Employee{id=3, name='Sergey, age=40, salary=7500}
Employee{id=2, name='Alex, age=28, salary=5500}
Employee{id=5, name='Maria, age=28, salary=3500}
```

GROUP BY Clause

- Условие Group By применяется для группировки собранных данных по какому-либо свойству объекта.

```
@Test
public void groupByTest() {
    EntityManager em = EMUtil.getEntityManager();
    javax.persistence.Query query = em.createQuery(
    "select count(e.name), e.name from Employee e group by e.name");
    query.getResultList().forEach(employees -> {
        Object[] emp = (Object[]) employees;
        System.out.println("Имя: " + emp[1] + " количество:" + emp[0]);
    });
}
```

```
HQL -> select count(employee0_.name) as col_0_0_, employee0_.name as col_1_0_ from
Employee employee0_ group by employee0_.name
Имя: Yulij количество:2
Имя: Sergey количество:1
Имя: Alex количество:1
Имя: Maria количество:1
```

Using Named Parameters

- Named Parameters используются для задания значения переменной в HQL-запрос.

```
@Test
public void parameterTest() {
    EntityManager em = EMUtil.getEntityManager();
    javax.persistence.Query query = em.createQuery(
    "from Employee e where e.name= :name");
    query.setParameter("name", "Yulij")
            .getResultList().forEach(System.out::println);
}
```

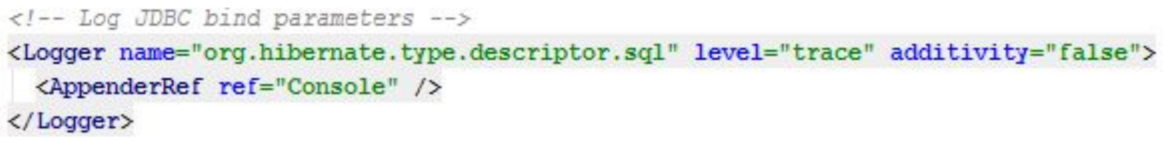
```
HQL -> select employee0_.id as id1_2_, employee0_.age as age2_2_, employee0_.name as name3_2_,
employee0_.salary as salary4_2_ from Employee employee0_ where employee0_.name=?
Employee{id=1, name='Yulij, age=30, salary=8500}
Employee{id=4, name='Yulij, age=40, salary=9500}
```

Using Named Parameters

- Named Parameters в порядке встречаемости

```
@Test
public void parameterOrderTest() {
    EntityManager em = EMUtil.getEntityManager();
    javax.persistence.Query query = em.createQuery(
            "from Employee e where e.name=? and e.salary >
:salary");
    query.setParameter(0, "Yulij")
            .setParameter("salary", 5000)
            .getResultList().forEach(System.out::println);
}
```

```
select employee0_.id as id1_6_, employee0_.age as age2_6_, employee0_.name as name3_6_, employee0_.salary as
salary4_6_ from Employee employee0_ where employee0_.name=? and employee0_.salary>?
HQL -> binding parameter [1] as [VARCHAR] - [Yulij]
HQL -> binding parameter [2] as [INTEGER] - [5000]
Employee{id=9, name='Yulij, age=30, salary=8500}
Employee{id=12, name='Yulij, age=40, salary=9500}
```



```
<!-- Log JDBC bind parameters -->
<Logger name="org.hibernate.type.descriptor.sql" level="trace" additivity="false">
  <AppenderRef ref="Console" />
</Logger>
```

Using Named Parameters

- Передача коллекции в качестве Named Parameters

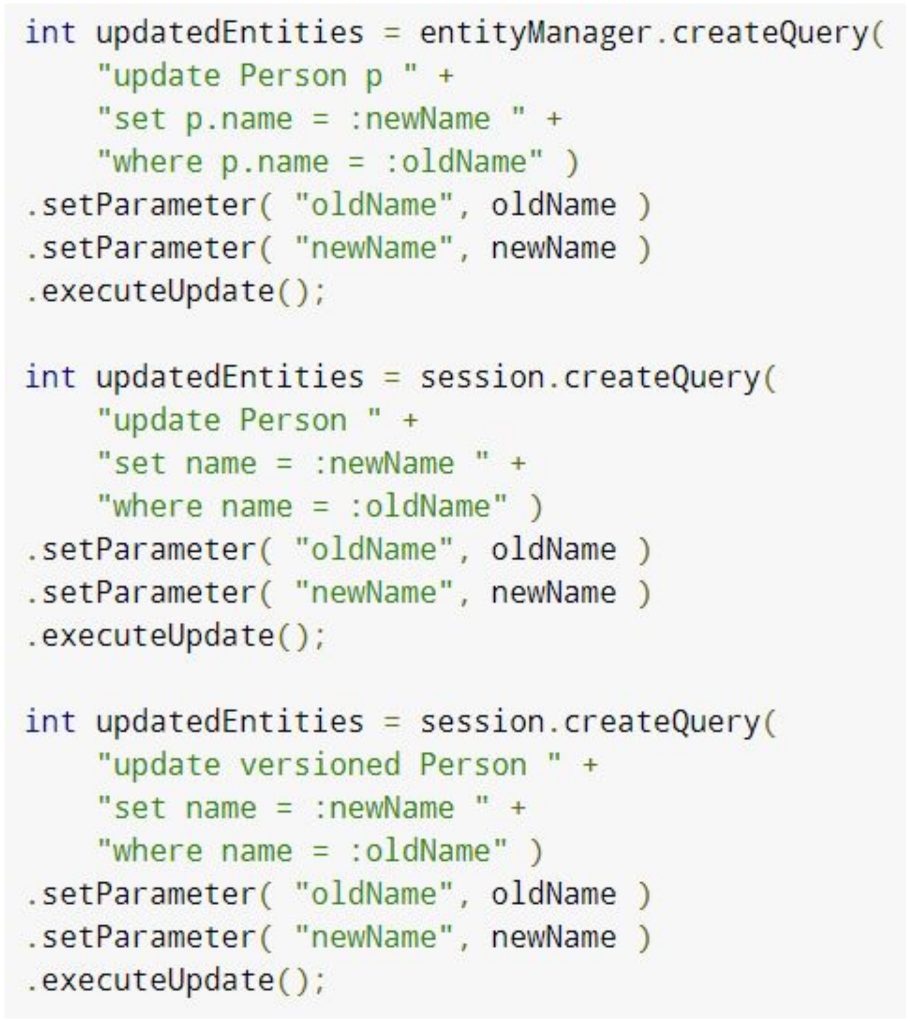
```
@Test
public void parameterListTest() {
    EntityManager em = EMUtil.getEntityManager();
    javax.persistence.Query query = em.createQuery(
     "from Employee e where e.id in(:ids)");
    query.setParameter("ids", Stream.of(1L,4L).collect(Collectors.toList()))
            .getResultList().forEach(System.out::println);
}
```

```
HQL -> select employee0_.id as id1_2_, employee0_.age as age2_2_, employee0_.name as name3_2_,
employee0_.salary as salary4_2_ from Employee employee0_ where employee0_.id in (? , ?)
Employee{id=1, name='Yulij, age=30, salary=8500}
Employee{id=4, name='Yulij, age=40, salary=9500}
```

Вопросы

UPDATE Clause

- Update применяется для обновления полей и свойств объектов в HQL.



```
int updatedEntities = entityManager.createQuery(
    "update Person p " +
    "set p.name = :newName " +
    "where p.name = :oldName" )
.setParameter( "oldName", oldName )
.setParameter( "newName", newName )
.executeUpdate();

int updatedEntities = session.createQuery(
    "update Person " +
    "set name = :newName " +
    "where name = :oldName" )
.setParameter( "oldName", oldName )
.setParameter( "newName", newName )
.executeUpdate();

int updatedEntities = session.createQuery(
    "update versioned Person " +
    "set name = :newName " +
    "where name = :oldName" )
.setParameter( "oldName", oldName )
.setParameter( "newName", newName )
.executeUpdate();
```

DELETE Clause

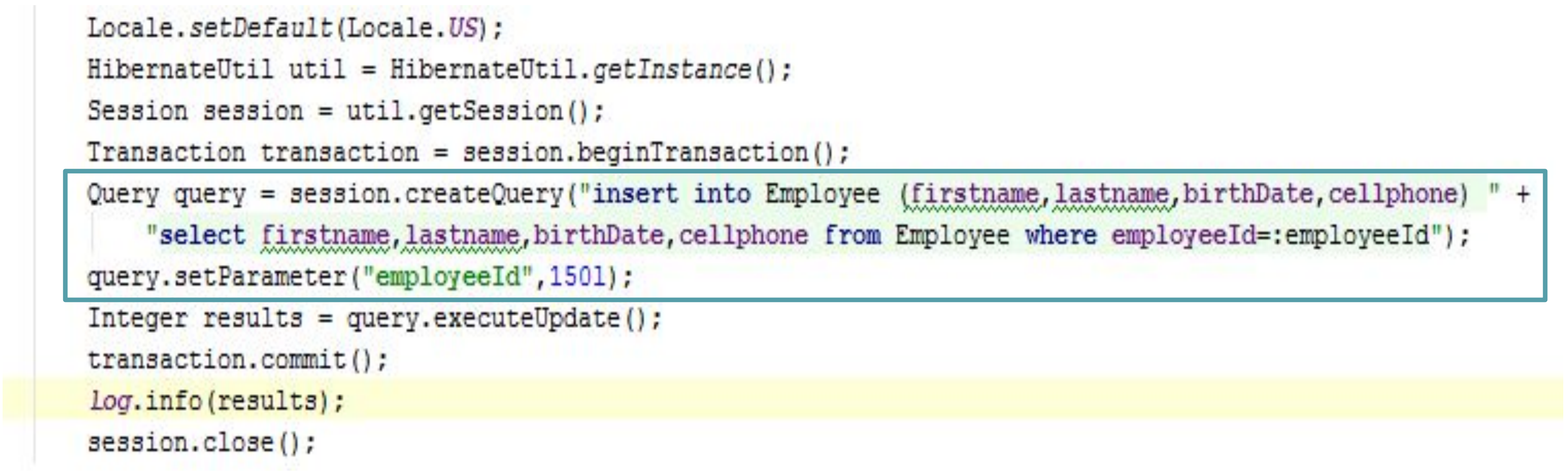
- Delete применяется для удаления одного или более объектов.

```
@Test
public void deleteTest() {
    EntityManager em = EMUtil.getEntityManager();
    Employee employee = new Employee(null, "Tuk", 100, 99);
    em.getTransaction().begin();
    em.persist(employee);
    javax.persistence.Query query = em.createQuery(
     "delete from Employee e where e.id=:id");
    System.out.println(
            query.setParameter( "id", employee.getId())
            .executeUpdate());
    em.getTransaction().commit();
}
```

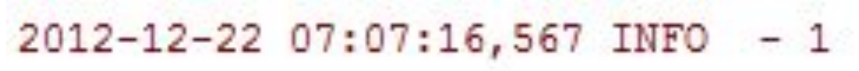
```
HQL -> insert into Employee (age, name, salary, id) values (?, ?, ?, ?)
HQL -> delete from Employee where id=?
1
```

INSERT Clause

- Insert применяется, когда нужно внести одну запись из другой, или другого объекта.



```
Locale.setDefault(Locale.US);
HibernateUtil util = HibernateUtil.getInstance();
Session session = util.getSession();
Transaction transaction = session.beginTransaction();
Query query = session.createQuery("insert into Employee (firstname,lastname,birthDate,cellphone) " +
    "select firstname,lastname,birthDate,cellphone from Employee where employeeId=:employeeId");
query.setParameter("employeeId",1501);
Integer results = query.executeUpdate();
transaction.commit();
log.info(results);
session.close();
```



```
2012-12-22 07:07:16,567 INFO  - 1
```

Вопросы

Aggregate Methods

HQL содержит ряд агрегационных функций:

- avg(property name)
- max(property name)
- min(property name)
- sum(property name)
- count(property name or *)
- count(...), count(distinct ...), count(all...)

Aggregate Methods

```
@Test
public void countDistinctTest() {
    EntityManager em = EMUtil.getEntityManager();
    javax.persistence.Query query = em.createQuery(
     "select count(distinct e.name), e.name from Employee e group by e.name");
    query.getResultList().forEach(employees -> {
        Object[] emp = (Object[]) employees;
        System.out.println("Имя: " + emp[1] + " количество:" + emp[0]);
    });
}
```

```
HQL -> select count(distinct employee0_.name) as col_0_0_, employee0_.name as col_1_0_ from Employee
employee0_ group by employee0_.name
Имя: Yulij количество:1
Имя: Sergey количество:1
Имя: Alex количество:1
Имя: Maria количество:1
```

Вопросы

Joins

```
@Test
public void joinTest() {
    EntityManager em = EMUtil.getEntityManager();
    List<Author> authors = em.createQuery(
            "select distinct a " +
                    "from Author a " +
                    "left join a.books b " +
                    "where b.title = 'War & Piece'", Author.class)
            .getResultList();
}
```

```
select distinct author0_.id as id1_0_, author0_.name as name2_0_ from Author author0_ left outer join Book books1_
on author0_.id=books1_.author_id where books1_.title='War & Piece'
HQL -> select books0_.author_id as author_i4_1_0_, books0_.id as id1_1_0_, books0_.id as id1_1_1_,
books0_.author_id as author_i4_1_1_, books0_.title as title2_1_1_, books0_.year as year3_1_1_ from Book books0_
where books0_.author_id=?
Author(id=1, name=Tolstoy, books=[
        Book{id=2, title='Alice', year=1872, author=Tolstoy},
        Book{id=3, title='War & Piece', year=1869, author=Tolstoy},
        Book{id=4, title='Philipok', year=1865, author=Tolstoy}
])
```

Join. WITH / ON

```
@Test
public void withJoinTest() {
    EntityManager em = EMUtil.getEntityManager();
    List<Author> authors = em.createQuery(
            "select distinct a " +
                    "from Author a " +
                    "inner join a.books b on b.title = 'War & Piece'")
            .getResultList();
    authors.forEach(System.out::println);
}
```

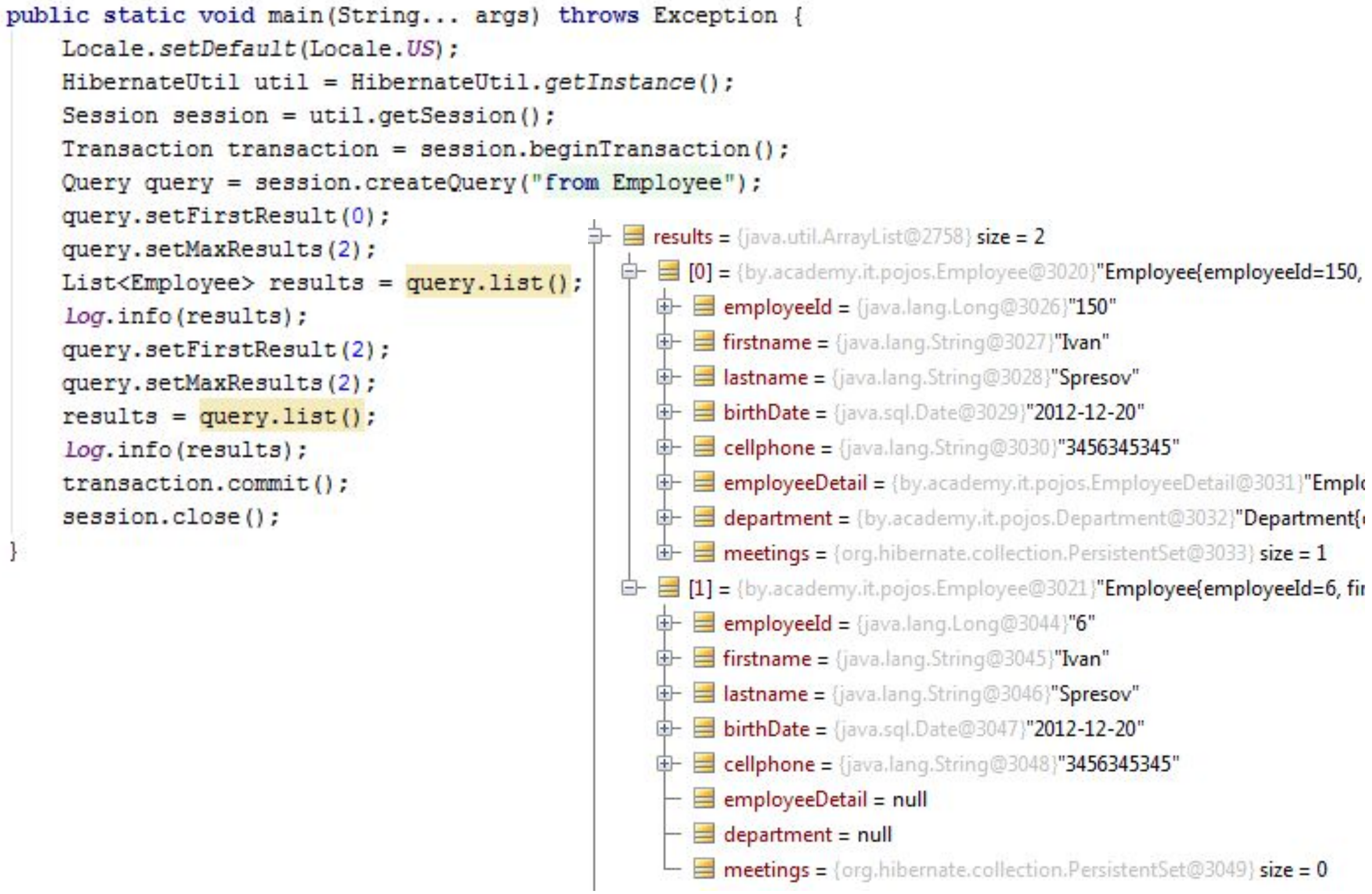
```
HQL -> select distinct author0_.id as id1_0_, author0_.name as name2_0_ from Author author0_
        inner join Book books1_ on author0_.id=books1_.author_id and (books1_.title='War & Piece')
HQL -> select books0_.author_id as author_i4_1_0_, books0_.id as id1_1_0_, books0_.id as id1_1_1_,
books0_.author_id as author_i4_1_1_, books0_.title as title2_1_1_, books0_.year as year3_1_1_ from Book
books0_ where books0_.author_id=?
Author(id=1, name=Tolstoy, books=[
Book{id=2, title='Alice', year=1872, author=Tolstoy},
Book{id=3, title='War & Piece', year=1869, author=Tolstoy},
Book{id=4, title='Philipok', year=1865, author=Tolstoy}])
```

Вопросы

Pagination using Query

- Постраничный вывод –это разбиение результата на страницы, т.е. на коллекции части ограниченного размера. Для пагинации в hibernate существуют следующие методы:
- **Query setFirstResult(int startPosition)**
- **Query setMaxResults(int maxResult)**

Pagination using Query



```
public static void main(String... args) throws Exception {
    Locale.setDefault(Locale.US);
    HibernateUtil util = HibernateUtil.getInstance();
    Session session = util.getSession();
    Transaction transaction = session.beginTransaction();
    Query query = session.createQuery("from Employee");
    query.setFirstResult(0);
    query.setMaxResults(2);
    List<Employee> results = query.list();
    log.info(results);
    query.setFirstResult(2);
    query.setMaxResults(2);
    results = query.list();
    log.info(results);
    transaction.commit();
    session.close();
}
```

```
results = {java.util.ArrayList@2758} size = 2
  [0] = {by.academy.it.pojos.Employee@3020}"Employee{employeeId=150,
    employeeId = {java.lang.Long@3026}"150"
    firstname = {java.lang.String@3027}"Ivan"
    lastname = {java.lang.String@3028}"Spresov"
    birthDate = {java.sql.Date@3029}"2012-12-20"
    cellphone = {java.lang.String@3030}"3456345345"
    employeeDetail = {by.academy.it.pojos.EmployeeDetail@3031}"Emplo
    department = {by.academy.it.pojos.Department@3032}"Department{
    meetings = {org.hibernate.collection.PersistentSet@3033} size = 1
  [1] = {by.academy.it.pojos.Employee@3021}"Employee{employeeId=6, fi
    employeeId = {java.lang.Long@3044}"6"
    firstname = {java.lang.String@3045}"Ivan"
    lastname = {java.lang.String@3046}"Spresov"
    birthDate = {java.sql.Date@3047}"2012-12-20"
    cellphone = {java.lang.String@3048}"3456345345"
    employeeDetail = null
    department = null
    meetings = {org.hibernate.collection.PersistentSet@3049} size = 0
```

Вопросы

Использование преобразователя в бин

```
import lombok.Data;
@Data
public class EmployeeWrapper {
    private Long id;
    private String firstName;
    private String password;
}
```

```
public List<EmployeeWrapper> setId(Long id) {
    return getSession().createSQLQuery("select e.id as id, e.first_name as
        firstName,e.password as password from Employee_History e
        where e.firstName = :name")
            .addScalar("id", StandardBasicTypes.LONG )
            .addScalar("firstName", StandardBasicTypes.STRING )
            .addScalar("password", StandardBasicTypes.STRING )
        .setParameter("name", employeeName)

.setResultTransformer(Transformers.aliasToBean(EmployeeWrapper.class))
            .list();
}
```

Вопросы

IT-Academy

Спасибо за внимание