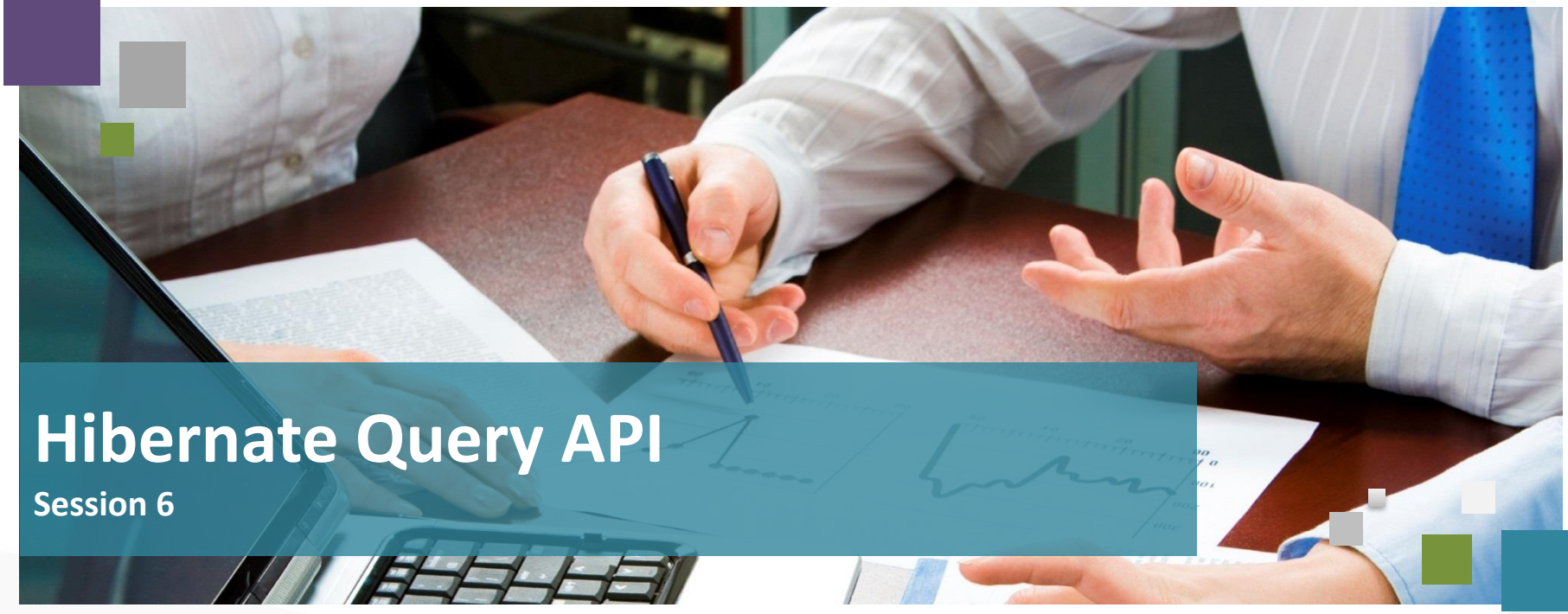
CitiusTech



Hibernate Query API

Session 6

Version 1.2

September 2018

Agenda

- **Introduction**
- Criteria Query
- Hibernate Query Language (HQL)
- Native Queries

Three Ways to Retrieving Data in Hibernate

- Criteria query API
  - The easiest way to retrieve data
  - Pure Java language based
- Hibernate Query Language (HQL)
- Native SQL query

Agenda

- Introduction
- **Criteria Query**
- Hibernate Query Language (HQL)
- Native Queries

Criteria Query

- Provides a set of Java objects for constructing queries
- Lets you build nested, structured query expressions in Java programming language
  - Compile time syntax checking possible
  - Polymorphic behaviour
- Supports Query By Example (QBE)
  - Performing a query by providing an example object that contain properties that need to be retrieved
- Supports aggregation methods (from Hibernate 3)
  - avg(...), sum(...), min(...), max(...)
  - count(*)
  - count(...), count(distinct ...), count(all...)

Usage of Criteria Query API

- The interface org.hibernate.Criteria represents a query against a particular persistent class
- The Session is a factory for Criteria instances

```
Criteria crit = sess.createCriteria(Cat.class);
crit.setMaxResults(50);
List cats = critlist();
```

- An individual query criterion is an instance of the interface **`org.hibernate.criterion.Criterion`**
- The class **`org.hibernate.criterion.Restrictions`** defines factory methods for obtaining certain built-in Criterion types

```
List cats = sess.createCriteria(Cat.class)
.add( Restrictions.like("name", "Fritz%") )
.add( Restrictions.between("weight", minWeight, maxWeight) )
.list();
```

Pagination

- Hibernate handles the pagination
  - Retrieving fixed number of objects
- Two methods of Criteria class
  - setFirstResult() - set the first row in the result
  - setMaxResults() - number of rows to retrieve

```
Criteria crit = sess.createCriteria(Person.class);
crit.setFirstResult(2);
crit.setMaxResults(50);
List results = crit.list();
```

Restrictions (1/4)

- You can narrow the result set via Restrictions Object
- Restrictions object is used to selectively retrieve objects
  - Person objects whose age is over 20
  - Add restrictions to the Criteria query object with add() method
  - The add() method of the Criteria object takes a org.hibernate.criterion.Criterion object that represents an individual restriction
- You can have more than one restriction for a Criteria query
- Methods of Restrictions class:

```
Restrictions.eq("name", "Shin")
Restrictions.ne("name", " NoName")
Restrictions.like("name", "Sa%")
Restrictions.ilike("name", "sa%")
Restrictions.isNull("name");
Restrictions.gt("price",new Double(30.0)) Restrictions.between("age", new
Integer(2), new Integer(10)) Restrictions.or(criterionl, criterion2)
Restrictions.disjunction()
```

Restrictions (2/4)

- Restrictions.like()

```
//Retrieve person objects whose name has a pattern
Criteria crit = sess.createCriteria(Person.class);
Criterion nameRestriction =Restrictions.like("name","Shin%");
crit.add( nameRestriction );
List results = crit.list();
```

- Restrictions can be logically grouped

```
// Retrieve Person objects whose name has a pattern and whose age is 10
 or null
List people = sess.createCriteria(Person.class)
.add( Restrictions.like("name", "Shin%") )
.add( Restrictions.or(
                    Restrictions.eq( "age", new Integer(10) ),
                    Restrictions.isNull("age"))
        )
.list();
```

Restrictions (3/4)

- There are a range of built-in criterion types (Restrictions subclasses). One of the most useful type allows you to specify SQL directly

```
List cats = sess.createCriteria(Cat.class)
    .add( Restrictions.sqlRestriction("lower({alias}.name) like lower(?)",
"Fritz%", Hibernate.STRING) )
    .list();
```

- The `{alias}` placeholder with be replaced by the row alias of the queried entity

Restrictions (4/4)

- You can also obtain a criterion from a Property instance. You can create a Property by calling `Property.forName()`:

```
Property age = Property.forName("age");
List cats = sess.createCriteria(Cat.class)
    .add( Restrictions.disjunction()
      .add( age.isNull() )
      .add( age.eq( new Integer(0) ) )
      .add( age.eq( new Integer(1) ) )
      .add( age.eq( new Integer(2) ) )
    ))
    .add( Property.forName("name").in( new String[] { "Fritz", "Izi", "Pk" } ) )
    .list();
```

Restrictions: Ordering the Results

- You can order the results using `org.hibernate.criterion.Order`

```
List cats = sess.createCriteria(Cat.class)
      .add( Restrictions.like("name", "F%")
      .addOrder( Order.asc("name") )
      .addOrder( Order.desc("age") )
      .setMaxResults(50)
      .list();
```

```
List cats = sess.createCriteria(Cat.class)
    .add( Property.forName("name").like("F%") )
    .addOrder( Property.forName("name").asc() )
    .addOrder( Property.forName("age").desc() )
    .setMaxResults(50)
    . list();
```

Projections (1/2)

- By navigating associations using createCriteria() you can specify constraints upon related entities:

```
List cats = sess.createCriteria(Cat.class)
    .add( Restrictions.like("name", "F%") )
    .createCriteria("kittens")
        .add( Restrictions.like("name", "F%") ).
    .list()
```

Projections (2/2)

- The projections concept is introduced in hibernate 3.0 and mainly we can do the following operations using the projection
  - **load partial object from the database**
  - **find the Result of Aggregate functions**
- The class **org.hibernate.criterion.Projections** is a factory for Projection instances. You can apply a projection to a query by calling `setProjection()`
- Projections class, has all static methods and each method of this class returns Projection interface object

```
List results = session.createCriteria(Cat.class)
      .setProjection( Projections.projectionList()
            .add( Projections.rowCount() )
            .add( Projections.avg("weight") )
            .add( Projections.max("weight") )
            .add( Projections.groupProperty("color") )
       )
       .list();
```

Query By Example (QBE)

- Provides another style of searching
- How to perform QBE based query
  - Partially populate an instance of an object
  - Hibernate will build a criteria using the instance as an example behind the scene
- **org.hibernate.criterion.Example** class implements Criterion interface
- You can use it like any other restrictions

```
// Retrieve person obects via example object

Criteria crit = sess.createCriteria(Person.class);
Person person = new Person();
Person.setName("Shin");
Example exampleRestriction = Example.create(person);
crit.add(exampleRestriction);
List result = crit.list();
```

Agenda

- Introduction
- Criteria Query
- **Hibernate Query Language (HQL)**
- Native Queries

Hibernate Query Language (HQL)

- Hibernate uses a powerful query language (HQL) that is similar in appearance to SQL
- HQL is fully object-oriented and understands notions like inheritance, polymorphism and association
- With the exception of names of Java classes and properties, queries are case-insensitive
- There is no need to qualify the class name since auto-import is the default

The Select Clause

- The select clause picks which objects and properties to return in the query result set

```
String SQL_QUERY _"Select
insurance.lngInsuranceId,insurance.insuranceName,insurance.investementAmount
,in surance.investementDate from Insurance insurance";

Query query = session.createQuery(SQL_QUERY);

for(Iterator it=query.iterate();it.hasNext();)
{
Object[] row = (Object[]) it.next();
System.out.println("ID: "+ row[0]);
System.out.println("Name: “+ row[1]);
 System.out.println("Amount: " row[2]);
}
```

The WHERE Clause

- **Where** clause is used to limit the results returned from database
- It can also be used with aliases

```
String SQL_QUERY =" from Insurance as insurance where
Insuranceing.InsuranceId='1";
 Query query = session.createQuery(SQL_QUERY);
for(Iterator it=query.iterate() it.hasNext();)
{ Insurance insurance=(Insurance)it.next();
  System.out.println("ID: "+ insurance.getLngInsuranceId());
  System.out.println("Name: "+ insurance.getInsuranceName());
}
```

The Group By Clause

- A query that returns aggregate values can be grouped by any property of a returned class or components

```
String SQL_QUERY = "select
sum(insurance.investementAmount),insurance.insuranceName from Insurance
insurance group by insurance.insuranceName";
Query query = session.createQuery(SQL_QUERY);
for (Iterator it = query.iterate(); it.hasNext();) {
Object() row = (Object()) it.next();
 System.out.println("Invested Amount: " + row[0]);
      System.out.println("Insurance Name: "+ row[1J);
                }
 String SQL_QUERY = "select avg(investementAmount) from Insurance insurance";
Query query = sess.createQuery(SQL_QUERY);
 List list = query.list();
System.out.println("Average of Invested Amount: "+ list.get(0));
```

The Order By Clause

- **Orderby** clause is used to order the entities based on specific entity property

```
Query query=session.createQuery("from Employee e order by e.sal desc");
List list=query.list();
Iterator iter=list.iterator();
while(iter.hasNext())
{
System.out.println(iter.next());
}
```

Agenda

- Introduction
- Criteria Query
- Hibernate Query Language (HQL)
- **Native Queries**

Native SQL Query

- Queries can be expressed in the native SQL dialect of the underlying database
- Execution of native SQL queries is controlled via the **SQLQuery** interface, which is obtained by calling `Session.createSQLQuery()`

```
public  SQLQuery createSQLQuery(String sqlString) throws HibernateException
```

- Code to read all the employees using Entity Query:

```
tx = session.beginTransaction();
String sql = "SELECT * FROM EMPLOYEE";
SQLQuery query = session.createSQLQuery(sql); query.addEntity(Employee.class);
List employees = query.list();
for (Iterator iterator = employees.iterator(); iterator.hasNext();)
{   Employee employee = (Employee) iteratornext();
 System.out.print("First Name: " + employee.getFirstName()); System.out.print("
Last Name: " + employee.getLastName()); System.out.println(" Salary: "+
employee.getSalary());  }
 tx.commit();
```

Thank You

CitiusTech Markets

CitiusTech Services

CitiusTech Platforms

Accelerating Innovation

CitiusTech Contacts

Email ct-univerct@citiustech.com

www.citiustech.com