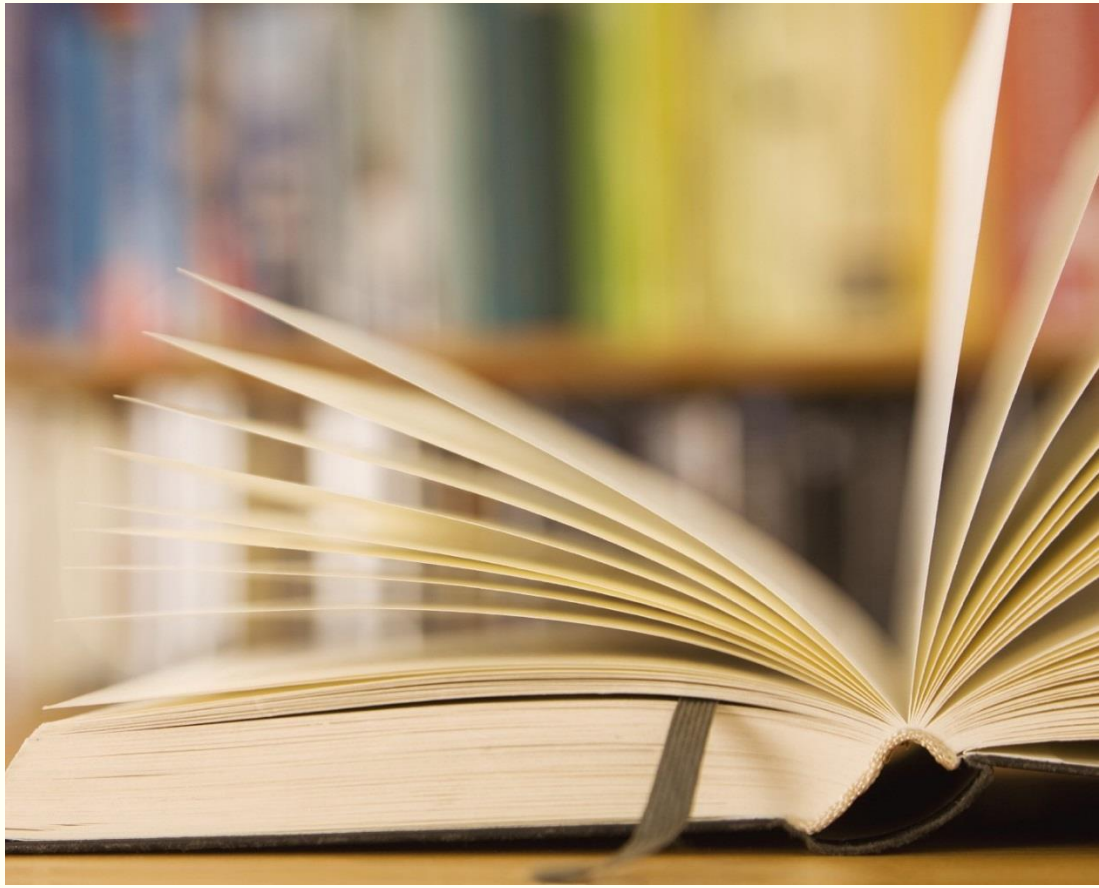
ENTERPRISE ARCHITECTURE

Najeeb Najeeb, PhD



CRITERIA API

Criteria API

- SELECT s FROM Student s

```
CriteriaBuilder criBuilder =
em.getCriteriaBuilder();

CriteriaQuery<Student> criQuery =
criBuilder.createQuery(Student.class);

Root<Student> rootStudent =
criQuery.from(Student.class);

criQuery.select(rootStudent);

TypedQuery<Student> query =
em.createQuery(criQuery);

List<Student> students
= query.getResultList();
```

Criteria API

- SELECT s FROM Student s WHERE s.gpa = 3.0

```
CriteriaBuilder criBuilder =
em.getCriteriaBuilder();

CriteriaQuery<Student> criQuery =
criBuilder.createQuery(Student.class);

Root<Student> rootStudent =
criQuery.from(Student.class);

criQuery.select(rootStudent);

criQuery.where(criBuilder.equal(rootStudent.
   get("gpa"), 3.0));

TypedQuery<Student> query =
em.createQuery(criQuery);

List<Student> students
= query.getResultList();
```

Joins

JPQL : "SELECT s FROM Student s WHERE s.gpa=3.0 AND s.laptop.ram= 8"

```
CriteriaBuilder criBuilder = em.getCriteriaBuilder();

CriteriaQuery<Student> criQuery = criBuilder.createQuery(Student.class);

Root<Student> rootStudent = criQuery.from(Student.class);

criQuery.select(rootStudent);

Predicate gpaPredicate = criBuilder.equal(rootStudent.get("gpa"), 3.0);

Join<Student, Laptop> joinLaptop = rootStudent.join("laptop");

Predicate ramPredicate = criBuilder.equal(joinLaptop.get("ram"), 8);

Predicate andPredicate= criBuilder.and(gpaPRediacte, ramPredicate);

criQuery.where(andPredicate);

TypedQuery<Student> query = em.createQuery(criQuery);

List<Student> students = query.getResultList();
```

Main Point

- With the increase in query complexity Criteria API can scale better. Criteria API also enables true type safety and compile time checking. This does reduce the control a developer has over the final queries that get generated. But it is always possible to configure the ORM to perform queries in certain ways.