buschmais

Beratung . Technologie . Innovation

Java User Group Saxony

Das JPA 2.0 Criteria-API

Frank Schwarz
buschmais GbR

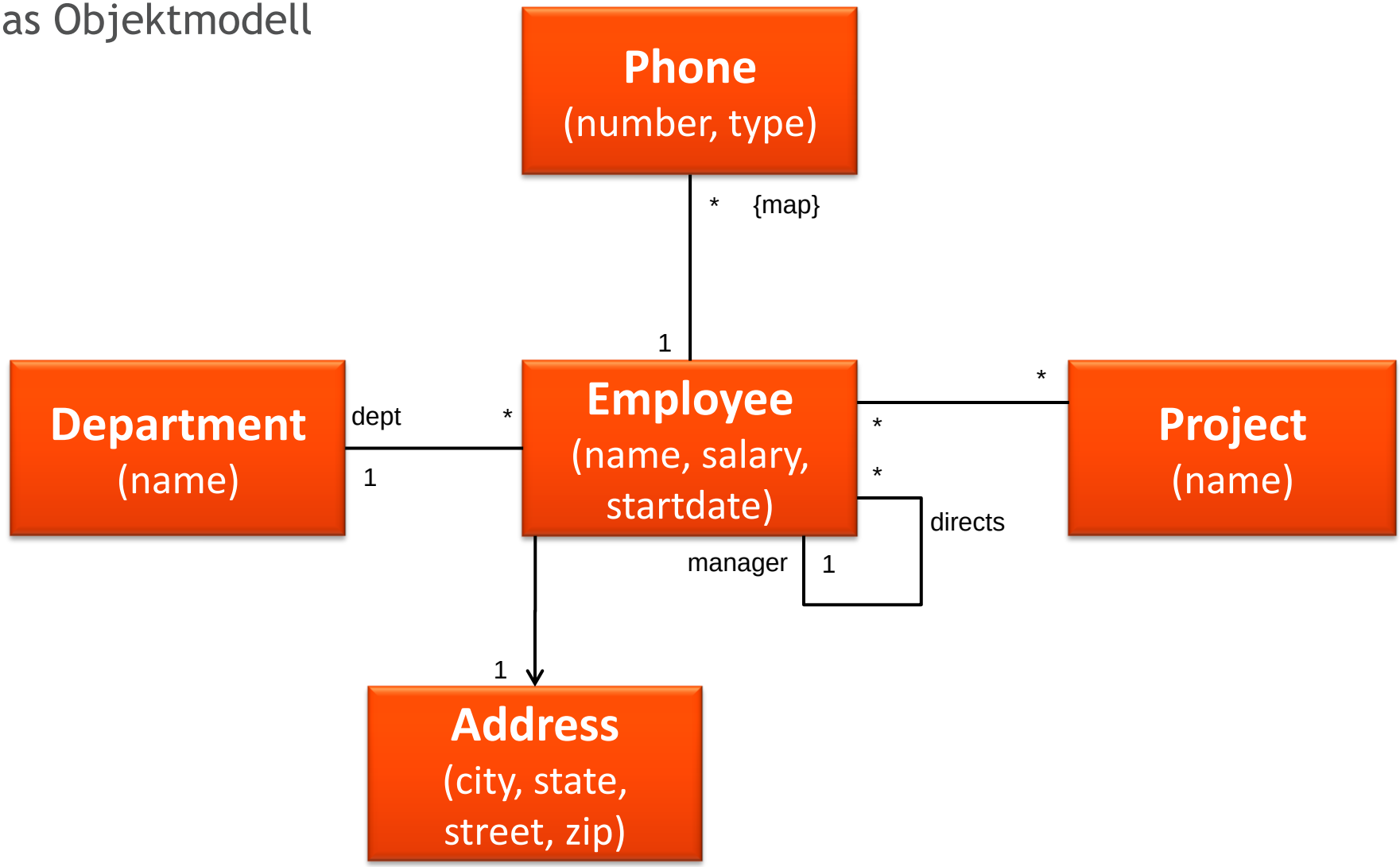
frank.schwarz@buschmais.de
http://www.buschmais.de/author/frank

Chemnitz, 24.06.2010

Das Criteria-API im Überblick

```
EntityManagerFactory emf = …

CriteriaBuilder cb = emf.getCriteriaBuilder();

EntityManager em = emf.createEntityManager()

CriteriaQuery<Candidate> query =
                    cb.createQuery(Candidate.class);

// Formulierung der Abfrage

List<Candidate> resultList =
              em.createQuery(query).getResultList();
```

Das Objektmodell



aus: Mike Keith, Merrick Schincariol: **Pro JPA 2: Mastering the Java™ Persistence API, 2009,** ISBN13: 978-1-4302-1956-9

Ergebnisse

Live-Coding

```java
public class SimpleQueryMain {

	public static void main(String[] args) {

		EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("demo");
		EntityManager entityManager = entityManagerFactory.createEntityManager();

		// Welche Abteilungen sind am Projekt 'Design Release2' beteiligt?
		System.out.println("-- JPQL ----------------------------------------");
		TypedQuery<Department> query1 = entityManager.createQuery(
				"select distinct d from Department d " +
				" join d.employees e join e.projects p " +
				" where p.name = 'Design Release2'", Department.class);
		List<Department> resultList1 = query1.getResultList();
		for (Department department : resultList1) {
			System.err.println(department);
		}

		System.out.println("-- Criteria -------------------------------------");
		CriteriaBuilder criteriaBuilder = entityManagerFactory.getCriteriaBuilder();
		CriteriaQuery<Department> query2 = criteriaBuilder.createQuery(Department.class);
		Root<Department> departmen1R = query2.from(Department.class);
		Join<Object, Object> employee1J = departmen1R.join("employees");
		Join<Object, Object> project1J = employee1J.join("projects");
		Predicate name1P = criteriaBuilder.equal(project1J.get("name"), "Design Release2");
		query2.select(departmen1R).where(name1P).distinct(true);

		List<Department> resultList2 = entityManager.createQuery(query2).getResultList();
		for (Department department : resultList2) {
			System.err.println(department);
		}

		System.out.println("-- Criteria mit Metamodell ----------------------");
		CriteriaQuery<Department> query3 = criteriaBuilder.createQuery(Department.class);
		Root<Department> department3R = query3.from(Department.class);
		SetJoin<Department, Employee> employee3J = department3R.join(Department_.employees);
		CollectionJoin<Employee, Project> project3J = employee3J.join(Employee_.projects);
		Predicate name3P = criteriaBuilder.equal(project3J.get(Project_.name), "Design Release2");
		query3.select(department3R).where(name3P).distinct(true);
		List<Department> resultList3 = entityManager.createQuery(query3).getResultList();
		for (Department department : resultList3) {
			System.err.println(department);
		}

		entityManager.close();
		entityManagerFactory.close();
	}
}
```

```
public class AggrgationQueryMain {

	public static void main(String[] args) {

		EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("demo");
		EntityManager entityManager = entityManagerFactory.createEntityManager();

		// Wieviele Mitarbeiter arbeiten in einer Abteilung?
		System.out.println("-- JPQL ----------------------------------------");
		TypedQuery<Object[]> query1 = entityManager.createQuery(
					"select d, count(e) " +
					" from Department d left join d.employees e" +
					" group by d" +
					" having count(e) > 1", Object[].class);

		List<Object[]> resultList1 = query1.getResultList();
		for (Object[] item : resultList1) {
			System.err.format("%s, %d%n", item);
		}

		System.out.println("-- Criteria -------------------------------------");
		CriteriaBuilder criteriaBuilder = entityManagerFactory.getCriteriaBuilder();
		CriteriaQuery<Tuple> query2 = criteriaBuilder.createTupleQuery();
		Root<Department> department2R = query2.from(Department.class);
		Join<Object, Object> employees2J = department2R.join("employees", JoinType.LEFT);
		Expression<Long> employee2CE = criteriaBuilder.count(employees2J);
		Predicate moreThan12P = criteriaBuilder.greaterThan(employee2CE, 1L);
		query2.multiselect(department2R, employee2CE)
			.groupBy(department2R)
			.having(moreThan12P);
		List<Tuple> resultList2 = entityManager.createQuery(query2).getResultList();
		for (Tuple tuple : resultList2) {
			System.err.format("%s, %d%n", tuple.get(department2R),
					tuple.get(employee2CE));
		}


		System.out.println("-- Criteria mit Metamodell ----------------------");

		entityManager.close();
		entityManager.close();

	}

}
```

```
public class AddressOfRobMain {

	public static void main(String[] args) {

		EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("demo");
		EntityManager entityManager = entityManagerFactory.createEntityManager();

		// Welche Adresse hat Rob?
		System.out.println("-- JPQL ----------------------------------------");
		TypedQuery<Address> query1 = entityManager.createQuery(
					"select a from Address a, Employee e " +
					" where e.name = 'Rob' " +
					" and e.address = a", Address.class);
		List<Address> resultList1 = query1.getResultList();
		for (Address address : resultList1) {
			System.err.println(address);
		}

		System.out.println("-- Criteria -------------------------------------");
		CriteriaBuilder criteriaBuilder = entityManagerFactory.getCriteriaBuilder();
		CriteriaQuery<Address> query2 = criteriaBuilder.createQuery(Address.class);
		Root<Address> address2R = query2.from(Address.class);
		Root<Employee> employess2R = query2.from(Employee.class);
		Predicate name2P = criteriaBuilder.equal(employess2R.get("name"), "Rob");
		Predicate join2P = criteriaBuilder.equal(employess2R.get("address"), address2R);
		Predicate where2P = criteriaBuilder.and(name2P, join2P);
		query2.select(address2R).where(where2P);
		List<Address> resultList2 = entityManager.createQuery(query2).getResultList();
		for (Address address : resultList2) {
			System.err.println(address);
		}


		entityManager.close();
		entityManager.close();

	}

}
```