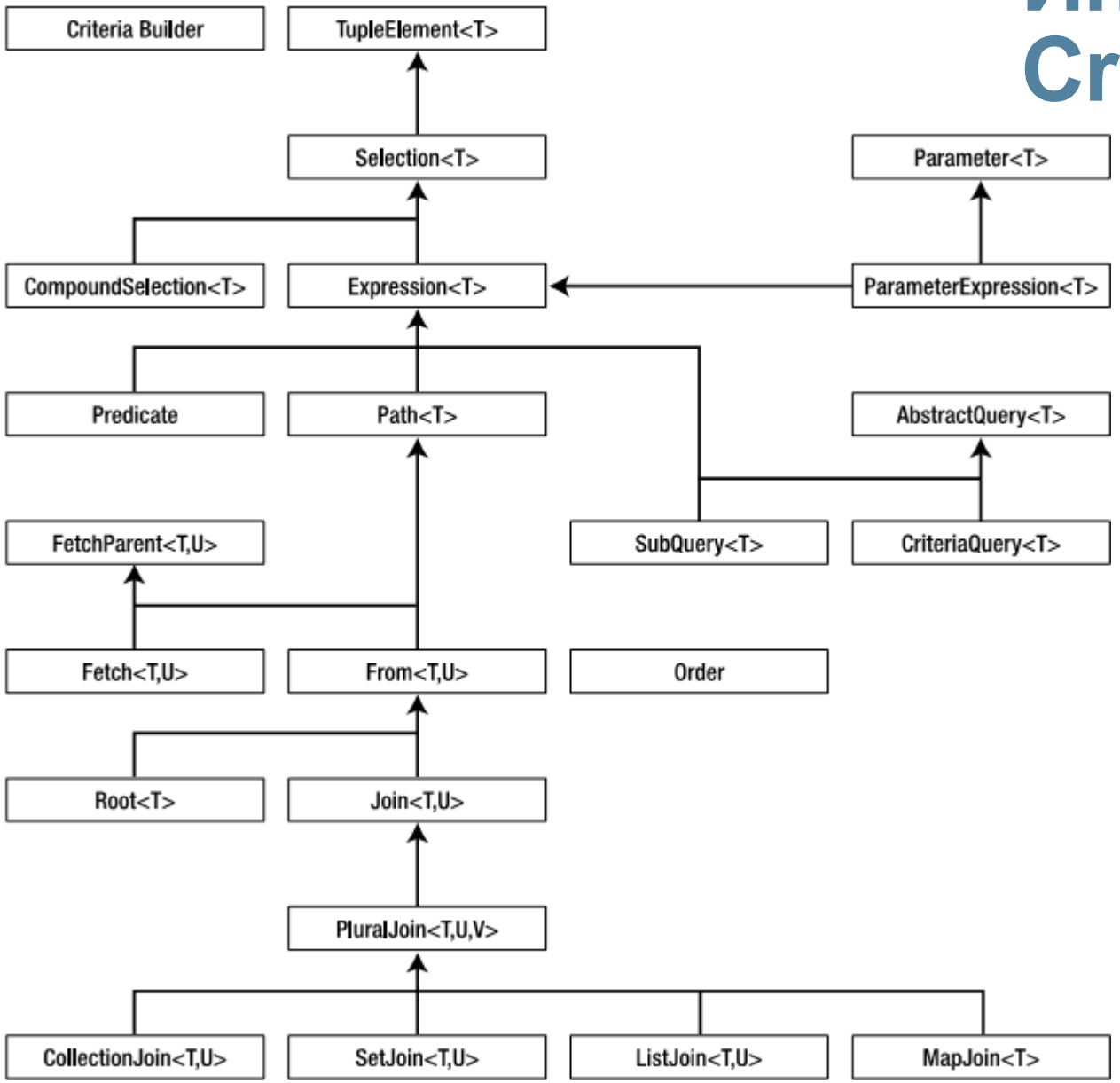
Запросы в JPA: Criteria API

Criteria API

- Появился в JPA 2.0
- Запрос **конструируется программно**
  - В JPQL-запросе легко допустить ошибку, которая будет выявлена только при запуске приложения
    - `SELEKT p FROM Person where p.age > 18`
  - Criteria API обеспечивает проверку во время разработки
    - В том числе type safety
    - Дополнение кода в IDE

Интерфейсы Criteria API



- Дублируют структуру JPQL-запроса

Динамические запросы

- В JPQL динамические запросы создаются конкатенацией строк

```
public List<Employee> findEmployees(String name, String deptName,
                                    String projectName, String city) {
    StringBuffer query = new StringBuffer();
    query.append("SELECT DISTINCT e ");
    query.append("FROM Employee e LEFT JOIN e.projects p ");

    query.append("WHERE ");
    List<String> criteria = new ArrayList<String>();
    if (name != null) { criteria.add("e.name = :name"); }
    if (deptName != null) { criteria.add("e.dept.name = :dept"); }
    if (projectName != null) { criteria.add("p.name = :project"); }
    if (city != null) { criteria.add("e.address.city = :city"); }
    for (int i = 0; i < criteria.size(); i++) {
        if (i > 0) { query.append(" AND "); }
        query.append(criteria.get(i));
    }

    Query q = em.createQuery(query.toString());
    if (name != null) { q.setParameter("name", name); }
    if (deptName != null) { q.setParameter("dept", deptName); }
    if (projectName != null) { q.setParameter("project", projectName); }
    if (city != null) { q.setParameter("city", city); }
    return (List<Employee>)q.getResultList();
}
```

Динамические запросы

- Criteria API

```
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Employee> c = cb.createQuery(Employee.class);
Root<Employee> emp = c.from(Employee.class);
c.select(emp);
c.distinct(true);
Join<Employee,Project> project =
    emp.join("projects", JoinType.LEFT);

List<Predicate> criteria = new ArrayList<Predicate>();
if (name != null) {
    ParameterExpression<String> p =
        cb.parameter(String.class, "name");
    criteria.add(cb.equal(emp.get("name"), p));
}
...
if (criteria.size() == 0) {
    throw new RuntimeException("no criteria");
} else if (criteria.size() == 1) {
   c.where(criteria.get(0));
} else {
    c.where(cb.and(criteria.toArray(new Predicate[0])));
}

TypedQuery<Employee> q = em.createQuery(c);
if (name != null) { q.setParameter("name", name); }
if (deptName != null) { q.setParameter("dept", deptName); }
if (project != null) { q.setParameter("project", projectName); }
if (city != null) { q.setParameter("city", city); }
return q.getResultList();
```

Type safe запросы в Criteria API

- Пример не-type safe запроса
  - ```
    query.where(qb.equal(employee.get("name"),
    "Bob"));
    ```
- Решение: каноническая метамодель для модуля персистентности
  - Генерируется инструментарием JPA-библиотеки
  - Позволяет избавиться от строковых констант
    ```

Type safe запросы в Criteria API

- Класс канонической метамодели

```
@StaticMetamodel(Employee.class)
public class Employee_ {
    public static volatile SingularAttribute<Employee, Integer> id;
    public static volatile SingularAttribute<Employee, String> name;
    public static volatile SingularAttribute<Employee, String> salary;
    public static volatile SingularAttribute<Employee, Department> dept;
    public static volatile SingularAttribute<Employee, Address> address;
    public static volatile CollectionAttribute<Employee, Project> project;
    public static volatile MapAttribute<Employee, String, Phone> phones;
}
```

- Генерация канонической метамодели
  - Пример для EclipseLink

```
javac -processor org.eclipse.persistence.internal.jpa.modelgen.CanonicalModelProcessor
        -proc:only
        -classpath lib/*.jar;punit
        *.java
```

- Type safe-запрос
  - `query.where(qb.equal(employee.get(Employee_.name), "Bob"));`