

- Java [[Object-Relational Mapping]] tool
- entity classes are defined via
 - annotations

Entity Classes

- Define persistent classes via annotations
- Add details for IDs, relationship types, and specific behavior on updates
- Some JPA implementations require enhancement process as post compilation step

```
@Entity
public class Student {
    @Id
    private int SID = -1;
    private String Fname;
    private String Lname;
    @ManyToMany
    private List<Course> ...
}
```

*

- [[XML]] files

Persistence Definition

- Separate XML meta data
META-INF/persistence.xml
- Includes connection details

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence.xml">
  <persistence-unit name="UniversityDB">
    <class>org.tugraz.Student</class>
    <class>org.tugraz.Course</class>
    <exclude-unlisted-classes/>
    <properties> ... </properties>
  </persistence-unit>
</persistence>
```

*

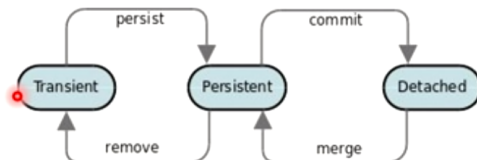
- object modification

CRUD Operations

- Insert by making objects persistent
- Update and delete objects according to object lifecycle states

Lifecycle States

- Lifecycle state transitions via specific persistence contexts
- Explicit and implicit transitions



```
EntityManager em = factory
    .createEntityManager();

tx.begin();

Student s = new
    Student(7,"Jane","Smith");
s.addCourse(new Course(...));
s.addCourse(new Course(...));

em.persist(s);

tx.commit();
em.close
```

- some ORM tools have special query languages
 - based on SQL

JPQL: Java Persistence Query Language

- SQL-like object-oriented query language
- Parameter binding similar to embedded SQL

```
EntityManager em = factory
    .createEntityManager();
Query q = em.createQuery(
    "SELECT s FROM Student s
     WHERE s.age > :age");
q.setParameter("age", 35);

Iterator iter = q
    .getResultList().iterator();
while( iter.hasNext() )
    print((Student)iter.next());
```

- programmatic APIs exist as well

JPQL Criteria API

- JPQL syntax and semantics with a programmatic API

```
.getResultList().iterator();  
while( iter.hasNext() )  
    print((Student)iter.next());
```

```
CriteriaQuery<Student> q = bld.createQuery(Student.class);  
Root<Student> c = q.from(Student.class);  
q.select(c).where(bld.gt(c.get("age"), bld.parameter(...)));
```

- sometimes native SQL queries still necessary
 - API not sufficient enough

JDBI

- Java Database Interface

Jdbi Overview

- Fluent API built on top of JDBC w/ same functionality exposed
- Additional simplifications for row to object mapping

Example

```
Jdbi jdbi = Jdbi.create("jdbc:postgresql://.../db1234567");  
Handle handle = jdbi.open();
```

```
jdbi.registerRowMapper(Student.class, (rs, ctx)  
    -> new Student(rs.getInt("sid"), rs.getString("lname"));
```

```
List<Student> ret = handle  
    .createQuery("SELECT * FROM Students WHERE LName = :name")  
    .bind(0, "Smith")  
    .map(Student.class)  
    .list();
```