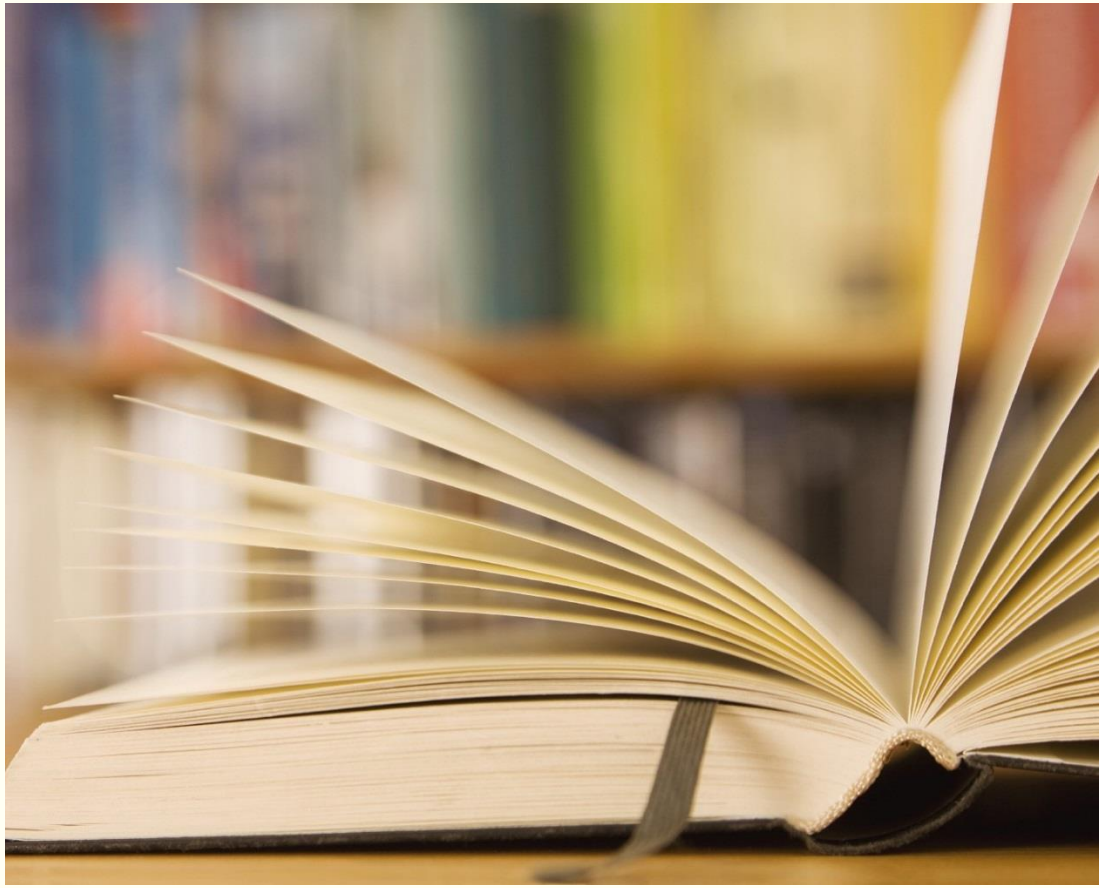
ENTERPRISE ARCHITECTURE

Najeeb Najeeb, PhD



SIMPLE ENTITIES

DATABASE PROBLEM

Our First Enterprise Problem

- Data outlives the process that creates it.
- An enterprise application deals with large amounts of data.
- Many options exist for storing data.
  - Relational Database Management Systems (RDBMS)
  - Files
- There are many hurdles to developing RDBMS software.
  - Having to write code to map objects to tables and columns (JDBC Dataset).
  - Write proper queries and ordering them to add/remove data (from related tables).
  - Having to make query optimizations.
  - Having to maintain complex SQL queries.
  - Keeping track of read data to prevent overloading the DB with queries.
  - Having to deal with data being split across several tables.

Ways to Work with DB

- Use proprietary solution (first one was Toplink).
- Use open-source solutions (first was Hibernate)
- Use Data Mappers (iBaties now MyBatis)
- Use JDBC
- Entity Beans (Enterprise JavaBeans)
- Java Data Objects (JDO) supported by OODB.
- A standard is needed, why
  - Not coupled to one product.
  - Not limiting an open-source project to a community.

Jakarta Persistence API - JPA

- Formally known as Java Persistence API.
- Jakarta EE application programming interface specification.
  - Describes the management of relational data in enterprise Java applications.
  - API
  - Jakarta Persistence Query Language JPQL (formally, Java Persistence Query Language)
  - Object/relational mapping.
- Object-relational Mapping – ORM
- JPA 1.0 (RI TopLink the former Oracle ORM)
- JPA 2.0 (RI EclipseLink)

ENTITY

What is an Entity

- Entities are not a new thing, existed before Java (Peter Chen paper "The entity-relationship model – toward a unified view of data" 1976.
- Things with attributes and relationships. Persisted in relational database.
- Noun, or grouped states.
- Participate in relationships with other entities.
- Persistable
  - Its state may be persisted.
  - Manipulated, can change state.
  - Application decides persistence.
- Identity
  - Object identity.
  - Persistent identity.

What is an Entity

- Transactional
  - Create, update, and delete are within a transaction.
- Granularity
  - Entities cannot be single-dimensional state.
  - Cannot be a single primitive type (or String).
  - Entities are meant to have relationships with other entities.
  - Business domain objects (generally).
  - Technically they could have one attribute or many.
    - But the intent is lightweight objects of average Java object.
- Metadata
  - Data that describe the entity.
  - Not persisted.
  - Stored internal to the class (annotation) or externally (XML).

Entity Contract

- POJO with a public or protected no-argument constructor.
- Not Final, no final methods or instance variables.
- Annotated with @Entity (javax.persistence.Entity).
- Implement's Serializable.
- May extend entity or non-entity classes.
- Instance variables must be private, protected or package.

First Entity

```
@Entity
public class Student {

        @Id private int id;
        private String name;
        private double gpa;

        public Student() {}
        public Student(int id) {  this.id = id; }

        @Id public int getId() {  return id; }
       public void setId(int id) {  this.id = id; }
       public Sttring getName() {  return name; }
       public void setName(String name) {  this.name = name; }
       public double getGpa() {  return gpa; }
       public void setGpa(double gpa) {  this.gpa = gpa; }

}
```

CRUD OPERATIONS

CRUD

- Create
- Read
- Update
- Delete
- EntityManager (javax.persistence.EntityManager) manages entaties.
- The set of managed entaties is the persistence context.
  - One Entity instance with a unique id exists in the persistence context.
- EntityManagerFactory (javax.persistence.EntityManagerFactory) creates EntityManager.
- Persistence unit hold the configuration needed by the EntityManagerFactory to create an EntityManager.

Create - Persist

- Instantiate a normal Java object.
- Use entity manager to persist the entity.
- What if things go wrong?
- Throw PersistenceException.

```
EntityManagerFactory emf=
Persistence.createEntityManagerFactory("MyPU");
EntityManager em = emf.createEntityManager();
Student jack = new Student(123, "Jack", 3.5);
em.persist(jack);
```

Read - Find

- EntityManager acts like a factory and returns the entity.
- Entity type is needed.
- Entity id is needed.
- The returned entity is managed.
- The method is typed.
- What if the entity is not found?

```
EntityManagerFactory emf=
Persistence.createEntityManagerFactory("MyPU");
EntityManager em = emf.createEntityManager();
Student jack = em.find(Student.class, 123);
```

Update

- There are several ways to update an entity.
- But only managed entities can be updated.
- If the entity is not managed, then the framework has no way of knowing about the change.

```
EntityManagerFactory emf=
Persistence.createEntityManagerFactory("MyPU");
EntityManager em = emf.createEntityManager();
Student jack = em.find(Student.class, 123);
jack.setGpa(3.2);
```

Delete - Remove

- Do you think most applications delete data?
- Only managed entities can be deleted.
- What will happen if you pass null to remove method of entity manager?

```
EntityManagerFactory emf=
Persistence.createEntityManagerFactory("MyPU");
EntityManager em = emf.createEntityManager();
Student jack = em.find(Student.class, 123);
em.remove(jack);
```

Transaction

- Do all operations need to be within a transaction?
- What is a good practice? Why?
- Javax.persistence.EntityTransaction
- Get the transaction from em.
- Mark the start and end of the transaction.
- Don't forget to provide rollback.

```
EntityTransaction tx = em.getTransaction();
tx.begin();
createStudent(123, "Jack", 3.0);
tx.commit();
```

Persistence Unit

- em from emf.
- emf from Persistence Unit.
- Stored in persistence.xml.

```
<persistence>
 <persistence-unit name="StudentService"
                   transaction-type="RESOURCE_LOCAL">
  <class> edu.miu.cs544.model.Student</class>
  <properties>
   <property name="javax.persistence.jdbc.driver"
             value= "org.apache.derby.jdbc.ClientDriver"/>
   <property name= "javax.persistence.jdbc.url"
             value= "jdbc:derby://localhost:1527/StudentDB;
             create=true"/>
   <property name="javax.persistence.jdbc.user"
             value= "userName"/>
   <property name= "javax.persistence.jdbc.password"
             value="password"/>
   </properties>
  </persistence-unit>
</persistence>
```

Main Point

- An entity is a POJO with a default constructor, it cannot be final, and it must have an identity (@Id). CRUD operations may be performed on an entity using entity manager.
- Everyone has a unique identity, with higher states of consciousness one can reach higher states and connect to the Self. The practice of TM and Sidhis enables one to reach higher states of consciousness.