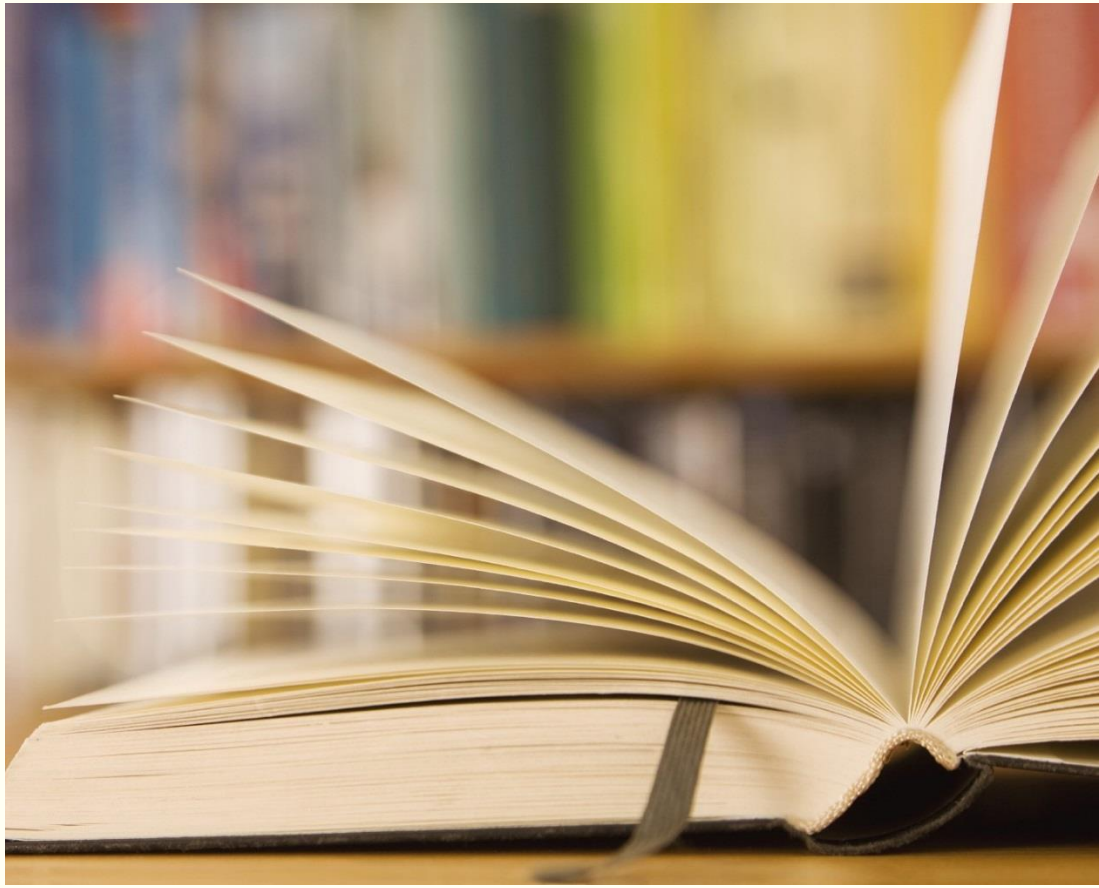
ENTERPRISE ARCHITECTURE

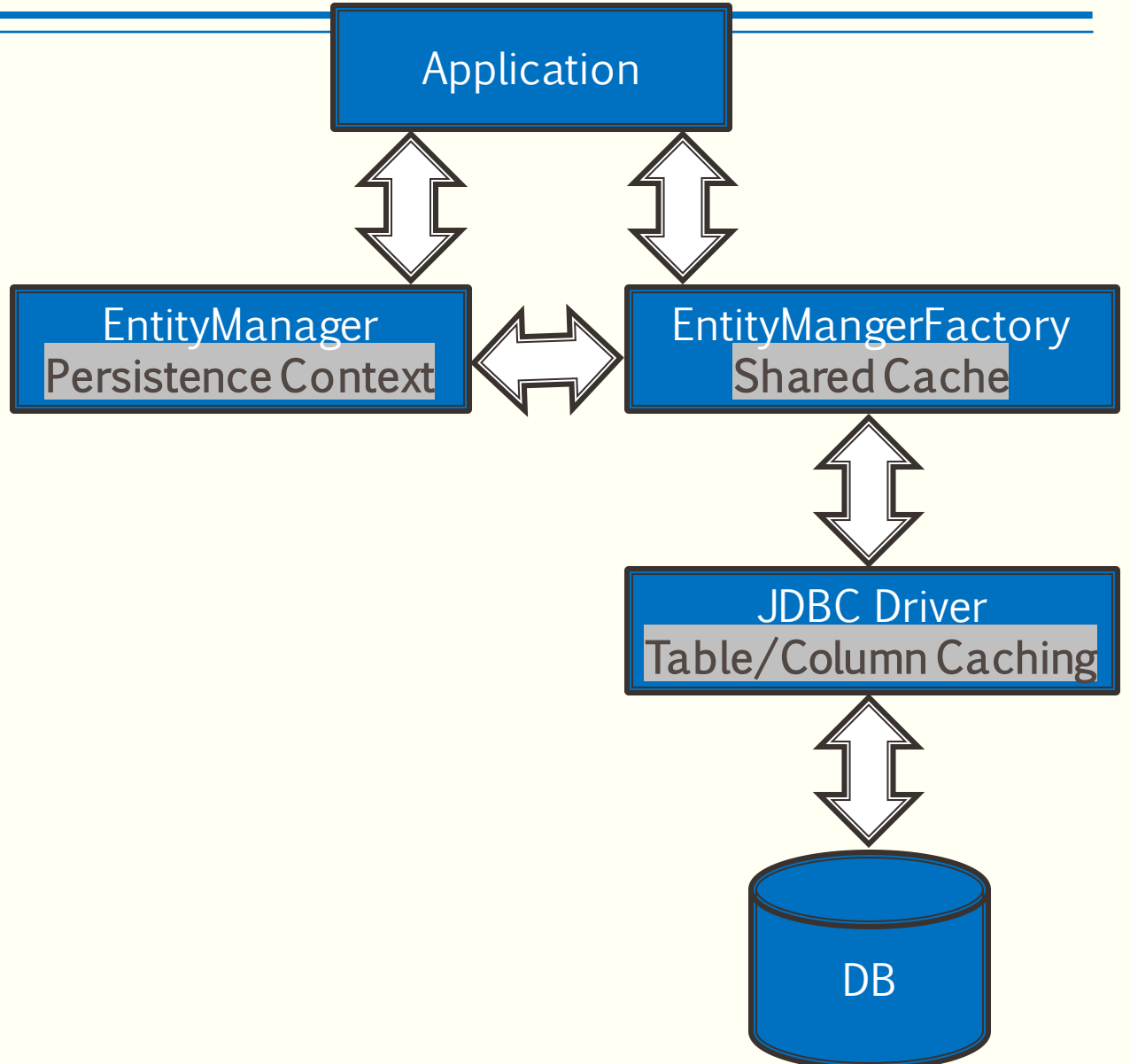
Najeeb Najeeb, PhD



CACHE

Layers of Cache

- Application cache
  - May get stale.
  - Discouraged.
- Persistence Contex
  - First level cache.
  - Reference heled by em.
  - Available while entity is attached.
  - One per em.
- Shared Cache
  - Second level cache (not always).
  - Shared by all ems from single emf.
- JDBC Cache
  - Driver level, unknown to JPA.



Shared Cache

- Cache Entities.
- Javax.persistence.Cache Interface
  - emf.getCache()
  - cache.contains()
  - cache.evictAll()
  - cache.evict(Entity.class, ID) , cache.evict(Entity.class)
- Should not be used in application code.
- Used in unit testing and for debugging.
- Clear cache between test cases, isolate tests.

Static Cache Configuration

- Level of configuration
  - Global, persistence unit.
  - Entity.
- persistence.xml or javax.persistence.sharedCache.mode in emf creation.
  - NOT_SPECIFIED (ORM default)
  - ALL
  - NONE
  - DISABLE_SELECTIVE (all entities are cached except @Cacheable(false))
  - ENABLE_SELECTIVE (all entities are not cached except @Cacheable)
- When not provided it is up to the ORM provider (default value).
- @Cacheable(false)

Cache Usage

- Read from Cache:
  - Select Queries.
  - em.find().
- Put in Cache:
  - Insert Queries.
  - em.find().
- If Cache is disabled, all methods return false (do nothing).
- Best practice
  - Keep Cache on and disable for relevant entities or queries.
  - You gain the ORM default (good) performance and address the issue you are facing.

Dynamic Cache Configuration

- Queries
  - setHint.
- EntityManager
  - Pass properties map.
- Read from Cache:
  - Property key javax.persistence.cache.retrieveMode
  - Property value CacheRetrieveMode.USE or CacheRetreiveMode.BYPASS
- Put in Cache:
  - Property key javax.persistence.cache.storeMode
  - Property value CacheStoreMode.USE or CacheStoreMode.BYPASS or CacheStoreMode.REFRESH
- REFRESH only needed when DB is written to by non-JPA or more than one JPA.

Dynamic Configuration Code

```
TypedQuery<Student> query=
em.createQuery("SELECT s FROM Student
s WHERE s.gpa > :gpa";

query.setHint("javax.persistence.cache.retr
iveMode", CacheRetreiveMode.BYPASS);

query.setHint("javax.persistence.cache.stor
eMode", CacheStoreMode.REFRESH);

query.setParameter("gpa", 3.0);

query.getResultList();
```

```
Map props= new HashMap();

props.put("javax.persistence.cache.retrieve
Mode", CacheRetrieveMode.BYPASS);

props.put("javax.persistence.cache.storeM
ode", CacheStoreMode.REFRESH);

em.find(Student.class, 5, props);
```

What About Refresh()?

- em.refresh(), make sure the context has the latest version (from DB).
- Update shared cache?
  - Use CacheStoreMode.REFRESH
- REFRESH can be set
  - Persistence unit.
  - Entity.

Main Point

- The Cache API refers to the EntityManagerFactory shared cache (Second Level Cache). You are not supposed to use the API in your application code. But if you find that caching is impacting your application you may demarcate cache behavior using static or dynamic configuration.