

Expense Manager Simple

Persistence Layer

PersistenceFactory

- Criar no package persistence a classe PersistenceFactory cujo objetivo é criar
 - A **Fábrica** de Fábricas de Repositórios, que é **única**.
O método buildRepositoryFactory() devolve a fábrica de repositórios de acordo com a estratégia escolhida. Isto é uma InMemoryRepositoryFactory (**old PersistenceRegistry**) ou JpaRepositoryFactory
- Os padrões utilizados:
 - **Factory**
 - **Singleton**
 - **Strategy**

```

public class PersistenceFactory {

    //SINGLETON
    private PersistenceFactory() {

        // ... ver a seguir
    }
    //LAZY LOADING - create only when needed
    private static PersistenceFactory instance = null;

    public static PersistenceFactory getInstance() {
        if (instance == null) {
            instance = new PersistenceFactory ();
        }
        return instance;
    }

    public IRepositoryFactory buildRepositoryFactory(){

        // ... ver a seguir
    }
    //....
}

```

Considere que existe na pasta raiz do projeto um ficheiro de propriedades **expensemanager.properties** Com a linha

PERSIST = persistence.JpaRepositoryFactory

PersistenceFactory Constructor

Considere que existe na pasta raiz do projeto um ficheiro de propriedades **expensemanager.properties** Com a linha

PERSIST = persistence.JpaRepositoryFactory

```

private PersistenceFactory() {
    //vai ao ficheiro config - Persist.config- buscar qual a
    //politica a usar . Por omissão hibernate
    try{
        FileInputStream propFile = new FileInputStream("Persistence.config");
        Properties p = new Properties(System.getProperties());
        p.load(propFile);
        System.setProperties(p);
        // p.propertyNames();
    } catch(Exception e) {
        System.setProperty("PERSIST", "JpaRepositoryFactory");
    }
    // System.setProperty("PERSIST", "persistence.JpaRepositoryFactory");
}
}

```

```

public IRepositoryFactory buildRepositoryFactory()
{
    String desc = System.getProperty("PERSIST");

    try
    {
        return (IRepositoryFactory) Class.forName(desc).newInstance();
    } catch (Exception ex)
    {
        return null;
    }
}

```

```

public class InMemoryRepositoryFactory implements
IRepositoryFactory{

    @Override
    public IExpenseRepository getExpenseRepository(){
        return new Persistence.inmemory.ExpenseRepository();
    }

    @Override
    public IExpenseTypeRepository getExpenseTypeRepository(){
        return new Persistence.inmemory.ExpenseTypeRepository();
    }
    @Override
    public IPaymentMeansRepository getPaymentMeansRepository(){
        return new Persistence.inmemory.PaymentMeansRepository();
    }
}

```

```

public class JpaRepositoryFactory
    implements IRepositoryFactory{

    @Override
    public IExpenseRepository getExpenseRepository(){
        return new persistence.jpa.ExpenseDAO();
    }

    @Override
    public IExpenseTypeRepository getExpenseTypeRepository(){
        return new persistence.jpa.ExpenseJpa();
    }

    @Override
    public IPaymentMeansRepository getPaymentMeansRepository(){
        return new persistence.jpa.PaymentMeansJpa();
    }

}

```

```

public interface IExpenseTypeRepository {

    public ExpenseType saveExpenseType(
        ExpenseType expenseType);

    public List<ExpenseType> getAllExpenseType();
}

```

Criar uma instância de um determinado Repositório

- Por exemplo:

- ExpenseRegisterController invoca o ExpenseRepository para gravar uma nova expense

```
public void saveExpense( Expense expense) {
```

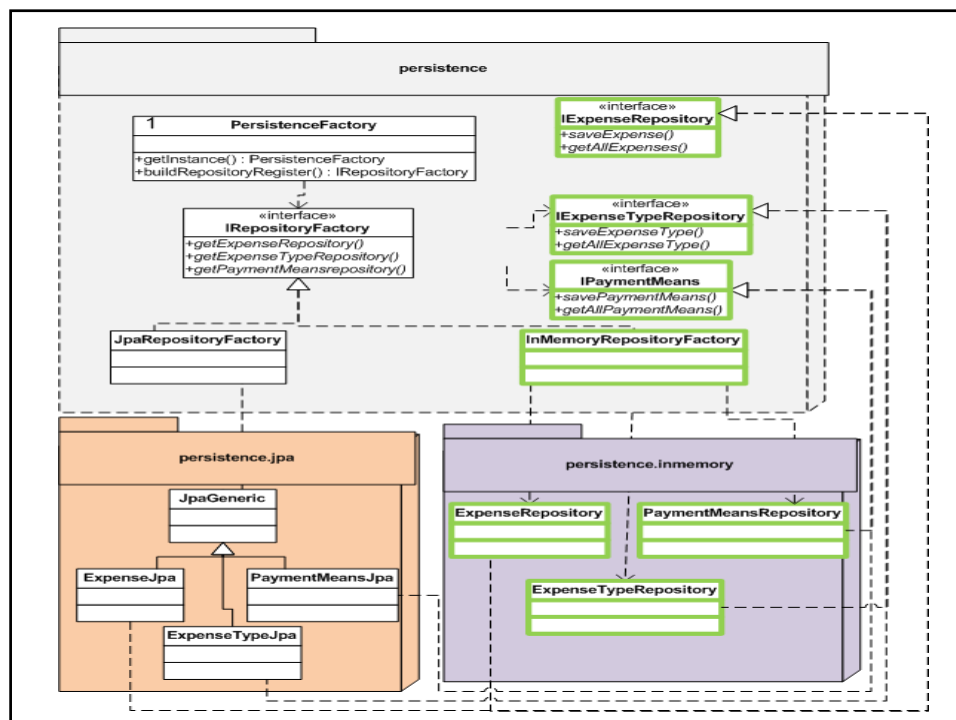
```
    IExpenseRepository repo=
```

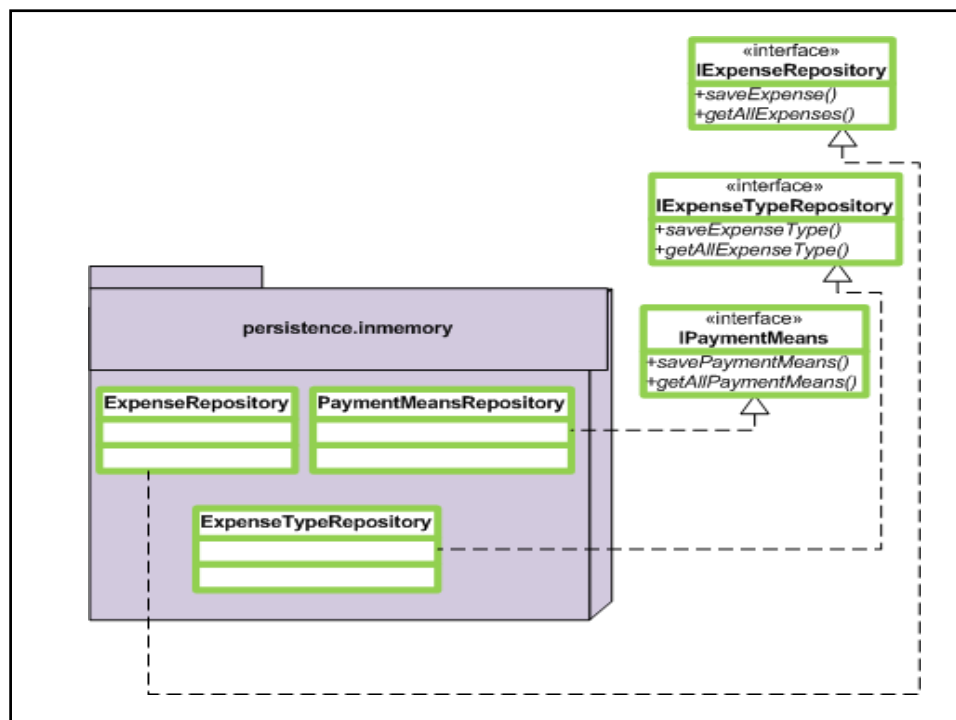
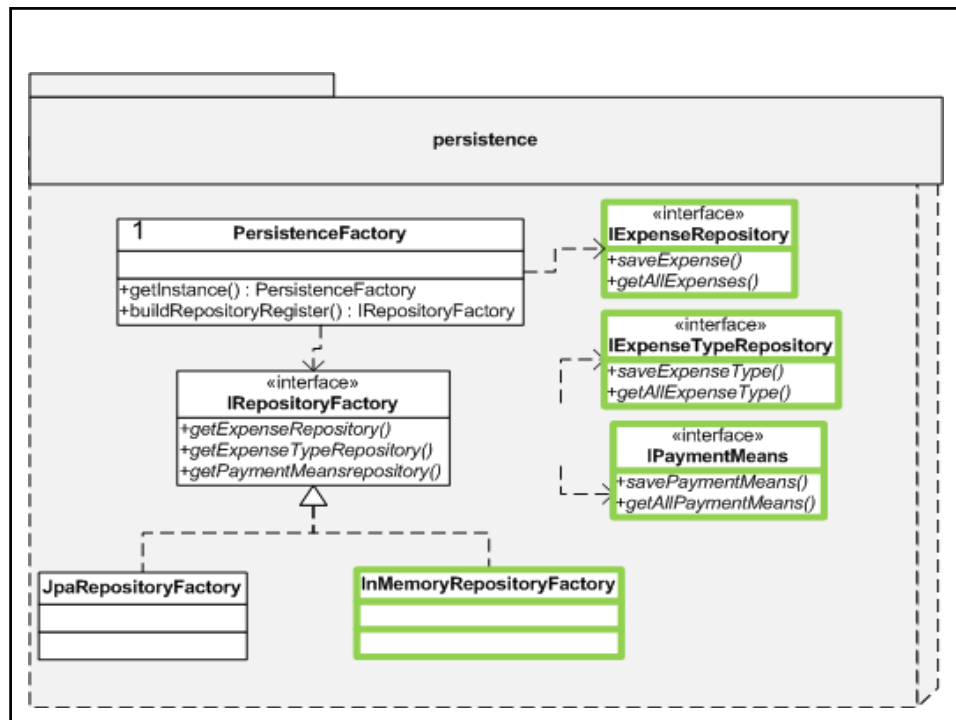
```
        PersistenceFactory.getInstance().buildRepositoryFactory.
```

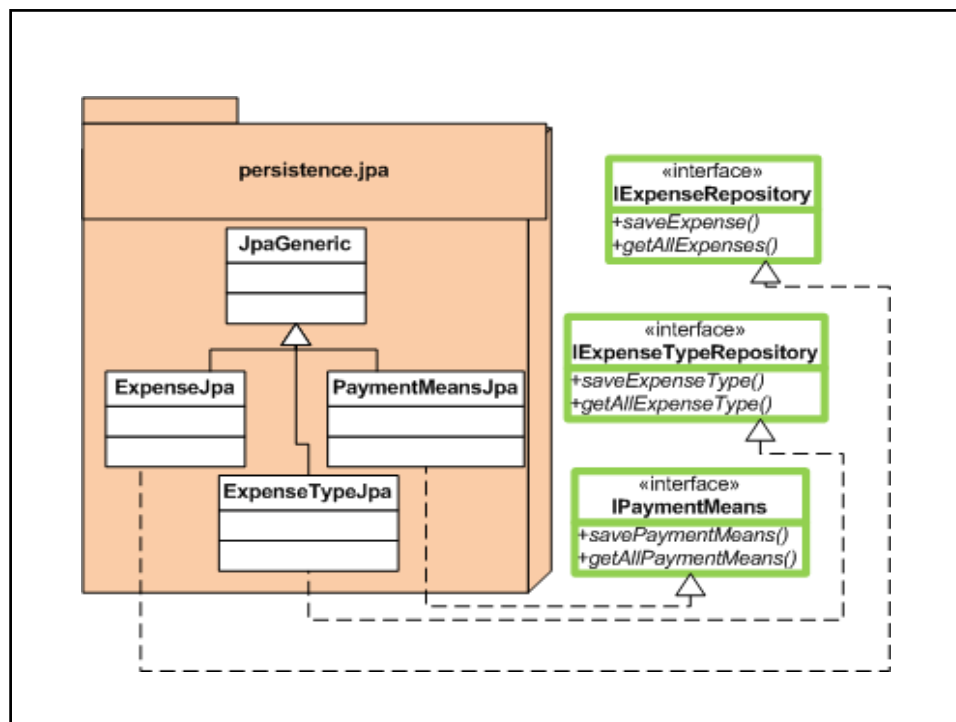
```
        getExpenseRepository();
```

```
        repo.saveExpense(expense);
```

```
    }
```







```

public abstract class JpaGeneric <T, PK extends Serializable> {

static protected EntityManagerFactory emf =
Persistence.createEntityManagerFactory("eapli.ExpenseManagerPU");

protected EntityManager getEntityManager() {
    EntityManager entityManager = emf.createEntityManager();
    return entityManager;
}

protected Class<T> entityClass;
public JpaGeneric() {
    ParameterizedType genericSuperclass =
    (ParameterizedType) getClass().getGenericSuperclass();
    this.entityClass = (Class<T>)
        genericSuperclass.getActualTypeArguments()[0];
}

```

```

public T create(T t) {
    this.getEntityManager().persist(t);
    return t;
}

public T read(PK id) {
    return this.getEntityManager().find(entityClass, id);
}

public T update(T t) {
    // Verificar se já existe
    return this.getEntityManager().merge(t);
}

public void delete(T t) {
    t = this.getEntityManager().merge(t);
    this.getEntityManager().remove(t);
}

```

```

public long getTotalCount() {
    return (Long) getEntityManager().
        createQuery("SELECT COUNT(*) FROM "
            + entityClass.getSimpleName() ).getSingleResult();
}

public Collection<T> findAll() {
    return getEntityManager().createQuery( "FROM " +
        entityClass.getSimpleName()).getResultList();
}

```



```

    * inserts or updates an entity
    * @param entity
    * @return the persisted entity - might be a different object than the parameter
    */
    public T save(T entity) {
        if (entity == null) {
            throw new IllegalArgumentException();
        }
        EntityManager em = getEntityManager();
        assert em != null;
        try {
            // transaction will be rolled back if any exception occurs
            EntityTransaction tx = em.getTransaction();
            try {
                tx.begin();
                em.persist(entity);
                tx.commit();
            } catch (PersistenceException ex) {
                tx.rollback();
                // we need to set up a new transaction if persist raises an exception
                tx = em.getTransaction();
                tx.begin();
                entity = em.merge(entity);
                tx.commit();
            }
        } finally {
            em.close();
        }
        return entity;
    }

```

```

public List<T> all() {

```

```

    EntityManager em = getEntityManager();
    assert em != null;

```

```

    String tableName = entityClass.getName();

```

```

    entityClass.getAnnotation(Table.class).name();

```

```

    Query q = em.createQuery("SELECT it FROM " +
        tableName + " it");

```

```

    List<T> all = q.getResultList();
    return all;
}

```