# Spring Boot

* @GeneratedValue annotation lets the framework to generate value and not by manually.
* Java class should have a public or protected non argument constructor.

## Entity Manager

* **Em.persist( obj )** is used to save or insert a record.
* **Em.merge( obj )** is used to update a record.
* When with entity manager you save a record in database with method **persisit()** and later on you changes any property of that object, that is also changed in database.

Course course = **new** Course(**"My Java Course"**);**entityManager**.persist(course);  
course.setName(**"My java course updated"**);

* **EntityManager.flush()** will send the data to database immediately and does not wait for transaction to commit.
* **EnitityManager.detach ( obj ):** By calling this method on an object, entity manager will stop tracking that object, whatever changes occurs to that object will not be persisted in database.
* **EntityManager.clear():** instead of calling detach method on each object, you call call this **clear()**  method and entity manager will stop tracking all changes occurring to all objects.
* **EntityManager.**refresh(): again start tracking

## Entity

* Entity is a pojo class , it represents a row in a table which we want to persist.

## Persistence Context

## CommandLineRunner vs ApplicationRunner

## @Transactional

This annotation is applied on repository class to take care of transactions.

When we manipulate data, perform any crud operation, that should be done via Transaction. A transaction takes care of **ACID** property of Database. We do not need transaction for find methods.

Transaction take care that if any error happens in performing crud operation, everything will be rolled back.

## Annotations

# Udemy Course: JPA and Hibernate with Spring Boot

# JPA: Java Persistence API

JPA says that you don’t need to worry about writing queries. Just tell me about your objects and how they map to your tables in database. Tell me about the relationships between tables and I’ll write queries on behalf of you.

## JDBC or Spring JDBC

Approaches present there before JPA. In which we were writing a lot of queries manually. Developer was responsible of writing right queries and giving right values in those queries.

## Entity

A uniquely identifiable object that we want to persist.

@Id, @GeneratedValue

* For @Id, we only generate getter method, because hibernate sets the value automatically and we don’t want others to set the value.
* Hibernate requires a no argument constructor.

## Repository

* In Repository, we add EntityManager

## JPQL :