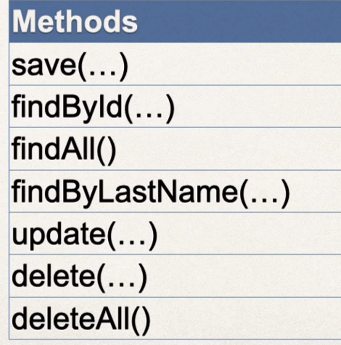
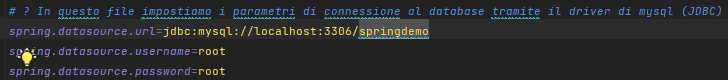
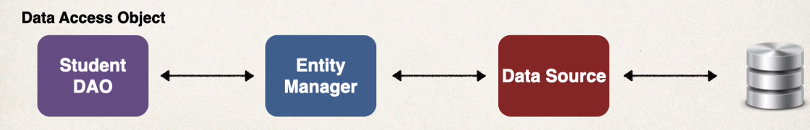
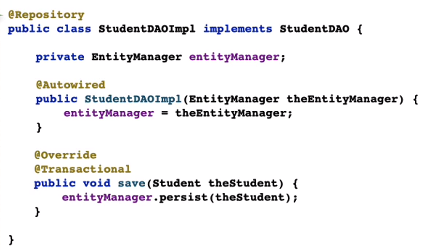
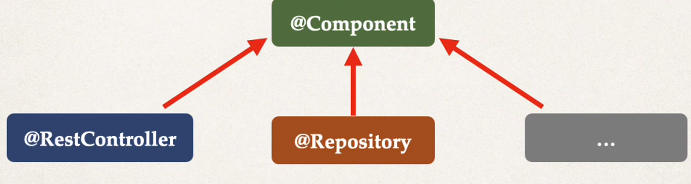
Data Access Object (DAO)  
  
In questo file trattiamo i concetti CRUD quindi vediamo come aggiungere un oggetto al database, come rimuoverlo, come aggiornare le proprietà di quest’oggetto e come leggerlo.  
  
Il DAO parlerà con il database, il dao è una normale classe o interfaccia java, la quale parla col db  
  
La nostra interfaccia DAO, avrà metodi che possono essere eseguiti su un database  
  
   
  
In questo scenario, il nostro oggetto di accesso ai dati, quindi colui che comunica con database, **ha bisogno** **di un gestore di entità JPA**.

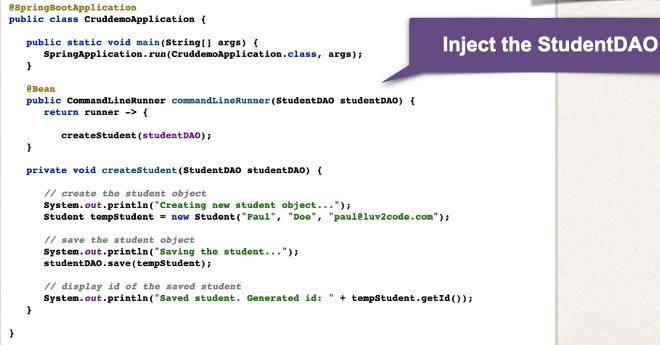
Ora il **GESTORE DI ENTITA JPA è il componente princiaple per eseguire quei metodi e quindi salvare, e recuperare le entità,** ciò vale a dire che il nostro oggetto DAO, utilizzerà un gestore di entità che ha altri componenti di supporto e poi comunicherà col database vero e proprio.  
  
Per quanto riguarda i componenti di supporto dobbiamo dire che:  
- Il nostro **entityManager ha bisogno di un’origine dati** → ciò significa che ha bisogno di dati di accesso al database **definiti,** **perchè in questo modo può stabilirne una connessione,** Questi dati vengo passati nel file application.properties come in questo caso:  
  
E quindi poi successivamente possiamo autocablare o iniettare il nostro GESTORE DI ENTITA JPA nel nostro oggetto DAO (student) → dicendo di iniettare un’instanza del gestore quando viene creata un instanza della classe che lo cabla  
  
  
Approfondimento gestore di entità → i**l gestore è una parte fondamentale di JPA**, serve come interfaccia tra l’applicazione java e il database. il compito principale del gestore e gestire il ciclo di vita degli oggetti persistenti, mantenere lo stato dell’oggetto e consentire lo scambio con il database.  
  
Il motivo per cui abbiamo bisogno del gestore è perchè JPA ci permette di gestire i nostri oggetti java come entità persistenti nel database. Infatti quando dobbiamo eseguire query al db come recupero o inserimento di un entità **dobbiamo fare riferimento all’utilizzo del gestore di entità**

**Quindi breve recap →** Abbiamo il nostro oggetto di accesso ai dati che utilizza un gestore di entità, il quale gestore utilizza a sua volta una determinata sorgente dati per connettersi al database, **e iniettermo il gestore di etità nella nostra INTERFACCIA dao, la quale dev’essere implementata!**

**  
  
{** Quindi nell’immagine abbiamo una classe Java che implementa l’interfaccia, la quale classe java utilizza un gestore di entità, il quale avvia una connessione col database ed è in grado di gestire i metodi di salvataggio e delete di un oggetto nel database, viene autocablato/iniettato nel costruttore della classe  
e viene sovrascritto un metodo dell’interfaccia, in fatti nel metodo **save** possiamo notare che viene richiamato l’oggetto di tipo GESTORE DI ENTITA PER SALVARE (persist) mettere dati nel database **}  
  
@Transactional →** spring offre questo tipo di annotazione, e ciò che facciamo aggiungendola sopra il nostro metodo **save,** in questo modo sappiamo che il nostro metodo viene eseguito all’interno di una transazione, **e quindi dietro le quinte spring si occuperà delle gestione delle transazioni**  
Cosa sono le transazioni nel contesto dei database? **→** una transazione è una sequenza di operazioni che devono essere eseguito come un’unita indivisibile, questo vale a dire che o tutte le opera<ioni vengono eseguite con successo e quindi i dati vengono salvati nel database o tutte vengono annullate e i dati tornano allo stato iniziale.  **Questa proprietà di transazione è fondamentale e mantine integro e coerente i dati nel database  
  
Quindi** ritornando all’annotazione **@Transactional** è un modo per dichiarare il comportamento transazionale di quel metodo o di una classe, e marcando quel metodo, stiamo dicendo a spring che tutte le operazioni all’interno di quel metodo devono essere eseguite all’interno di una transazione, per il motivo che anche se un solo dato non è corretto viene annullata la transazione.

**Questa notazion fa parte del framework Spring e offre per i dao alcune annotazioni importanti, tra cui: DAO NOTATION:  
  
- Repository → viene utilizzato per annotare i DAO è possiamo considerarla una sotto annotazione   
  
**

Quindi marcata una **classe** con questa notazione, spring scansionando tutti i suoi componenti registrerà automaticamente l’implementazione quindi capirà che si tratta di una repository

**BONUS →** Ora nel nostro commandLiner dobbiamo iniettare il nostro oggetto DAO.  
  


Tutto definito in STEP → consapevoli del fatto che abbiamo già una classe entity che rappresenta la tabella.  
  
Step 1) Definisci l’interfaccia DAO

Step 2) Definisci l’implementazione dell’interfaccia, creando nel package dao, una classe chiamata studentDAOimpl che indica che quella classe ha implementato l’interfaccia studentDAO, e aggiungi la notazione **@Repository alla classe → indiacando che questa è una classe di accesso ai dati, e viene utilizzata per gestire la persistenza dei dati relativi agli oggetti student**Step 3) implementiamo e definiamo il metodo che dovrà essere sovrascritto e non dimentichaiamoci di marcare questo metodo con la notazione transactional in modo che spring tratterà questo metodo con un comportamento transazionale, creiamo il nostro Gestore di entità e creeremo anche il nostro costruttore il quale accetta come argomento il gestore, e sarà fatta un’iniezione del gestore ogni volta che sarà richiamata la classe