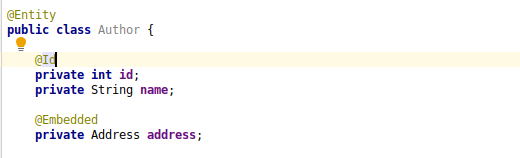
**Spring Data JPA with Hibernate Part 3**

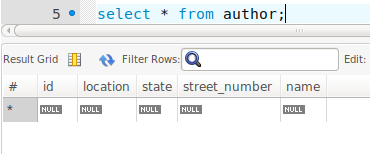
1. **Create a class Address for Author with instance variables streetNumber, location, State.**

****

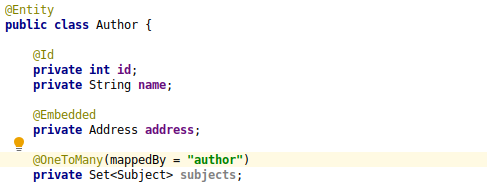
1. **Create instance variable of Address class inside Author class and save it as embedded object.**

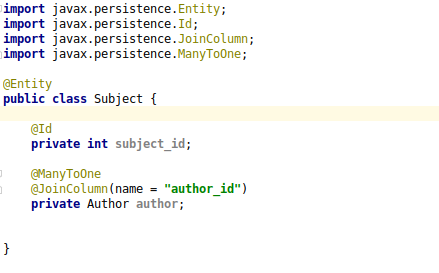
****

Author table after embbeded.

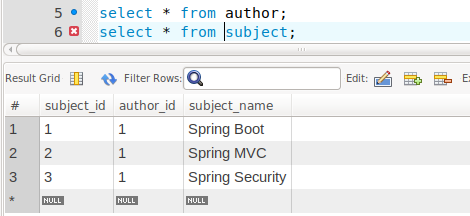
****

1. **Introduce a List of subjects for author.**

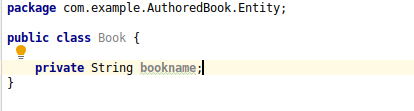
****

****

1. **Persist 3 subjects for each author.**

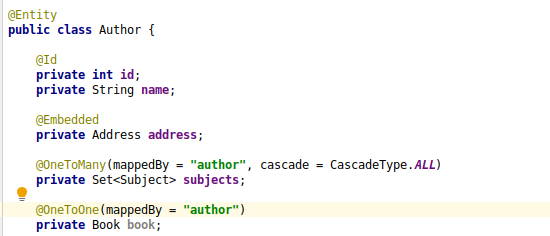
****

1. **Create an Entity book with an instance variable bookName.**

****

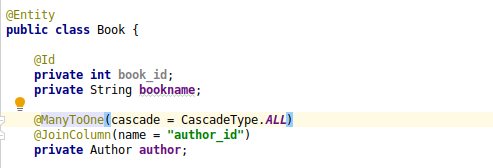
1. **Implement One to One mapping between Author and Book.**

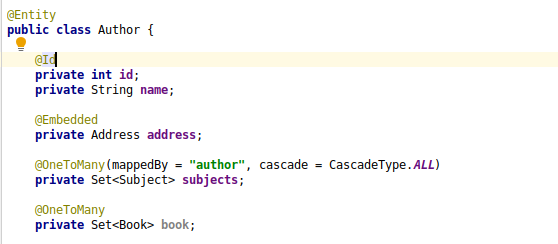
****

****

1. **Implement One to Many Mapping between Author and Book(Unidirectional, BiDirectional and without additional table ) and implement cascade save.**

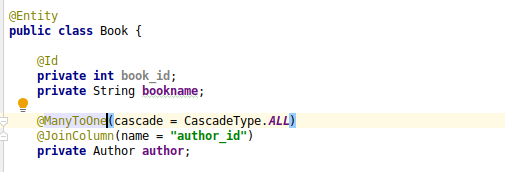
Unidirectional:





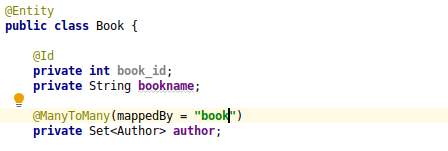
Bidirectional:



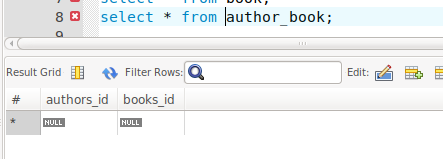


1. **Implement Many to Many Mapping between Author and Book.**

****

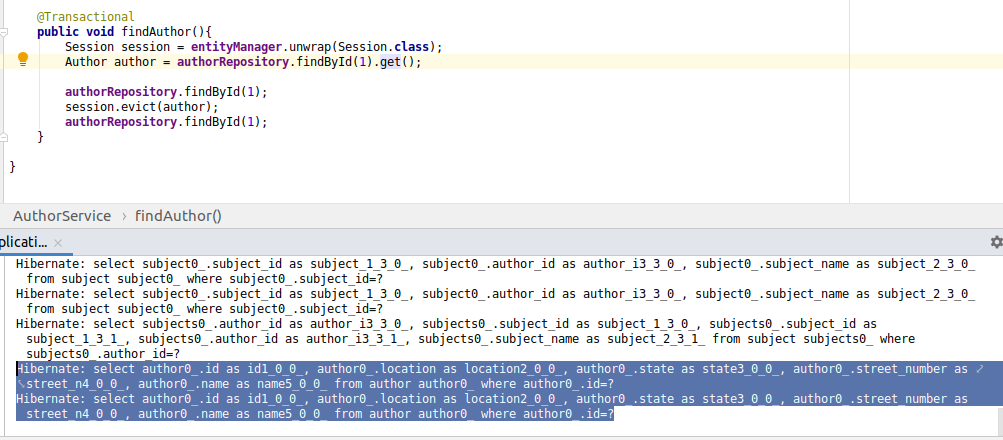
****

New table created:



1. **Which method on the session object can be used to remove an object from the cache?**

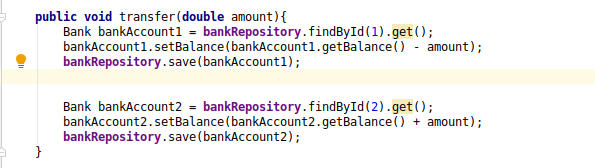
If we evict an session object that query will be removed from cache and will perform the select query again to fetch the data.

****

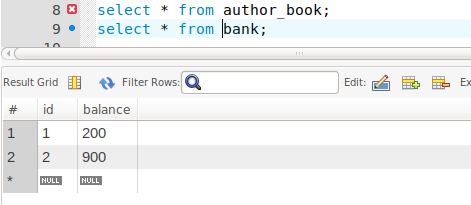
1. **What does @transactional annotation do?**

@Transactional is used to maintain the atomicity of a transaction. It means either all transactions will happen or nothing will happen.

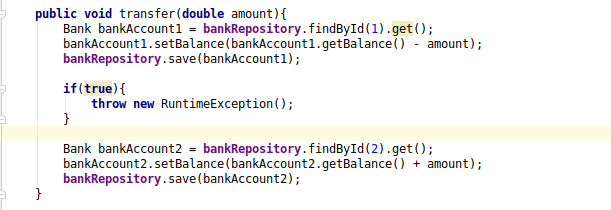
Taking an example we created 2 bank objects with starting balance: 500 & 600. Now if we perform normal transaction action it will perform without any problem.



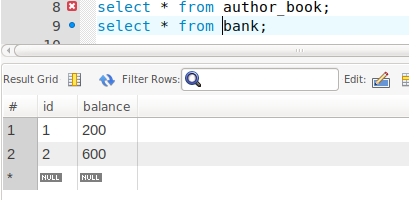
We took 500 from id 1 and added it to id 2 amount. So total after nad before transactions will be same.



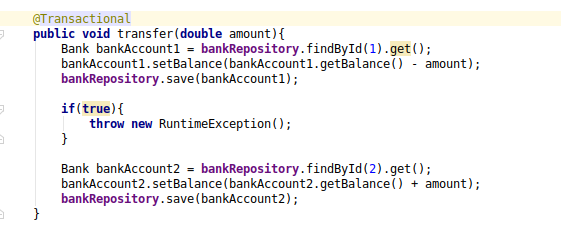
But what happens if there will be any error occurring after the first transaction. Then the first transaction will happen and the second will not.



Here 500 is deducted from the first id1 but due to error the statement didn’t reach the 2nd transaction. So the integrity is lost.



So to avoid this we use @Transactional annotation. It ensures that either all transaction happen or nothing happens at all.



Here @Transactional rolled back the first transaction after the second transaction didn’t reached thus maintaining the integrity of database.

