Saturday, 24 August 2024

Hibernate Cache

Subject

- **First Level Cache Important Points:**

1. Hibernate First Level cache is enabled by default, there are no configurations needed for this.
2. Hibernate first level cache is session specific, that's why when we are getting the same data in same session there is no query fired whereas in other session query is fired to load the data.
3. Hibernate first level cache can have old values, as you can see above that I have put my program to sleep for 10 seconds and in that time I updated the value (name from Pankaj to PankajK) in database but it didn't get reflected in the same session. But in other session, we got the updated value.
4. We can use session evict() method to remove a single object from the hibernate first level cache.
5. We can use session clear() method to clear the cache i.e delete all the objects from the cache.
6. We can use session contains() method to check if an object is present in the hibernate cache or not, if the object is found in cache, it returns true or else it returns false.
7. Since hibernate cache all the objects into session first level cache, while running bulk queries or batch updates it's necessary to clear the cache at certain intervals to avoid memory issues.

- **Second Level Cache**

8. Read Only: This caching strategy should be used for persistent objects that will always read but never updated. It's good for reading and caching application configuration and other static data that are never updated. This is the simplest strategy with best performance because there is no overload to check if the object is updated in database or not.
9. Read Write: It's good for persistent objects that can be updated by the hibernate application. However if the data is updated either through backend or other applications, then there is no way hibernate will know about it and

data might be stale. So while using this strategy, make sure you are using Hibernate API for updating the data.

10. Nonrestricted Read Write: If the application only occasionally needs to update data and strict transaction isolation is not required, a nonstrict-read-write cache might be appropriate.
11. Transactional: The transactional cache strategy provides support for fully transactional cache providers such as JBoss TreeCache. Such a cache can only be used in a JTA environment and you must specify hibernate.transaction.manager_lookup_class.

    1.