Persistence Context

https://medium.com/@nandaras0103/java-persistence-api-52c6731608dd

https://docs.oracle.com/javaee/7/tutorial/persistence-intro003.htm#BNBQW

JPQL

The JPQL (Java Persistence Query Language) statement you provided is a query that retrieves `Humanoid` entities based on a condition involving a collection-valued association attribute (`outfits`). Let's break down the query:

```sql
select h from Humanoid h
where :outfit member of h.outfits
```

- `select h from Humanoid h`: This part of the query selects all `Humanoid` entities (`h`) from the database. `h` is an alias for the `Humanoid` entity in the query, allowing us to reference its attributes and properties.

- `where :outfit member of h.outfits`: This is the WHERE clause of the query. It specifies a condition that filters the selected `Humanoid` entities based on a specific condition involving the `outfits` association attribute.

  - `:outfit` is a named parameter in the query. Named parameters start with a colon (`:`) followed by a parameter name (`outfit`). Named parameters are placeholders that can be set with actual values when executing the query.

  - `member of` is a JPQL operator that checks whether the specified value (`:outfit`) is a member of a collection. In this case, it checks whether the value of the `outfits` collection attribute of each `Humanoid` entity (`h`) contains the specified `outfit` parameter.

The overall meaning of the query is to retrieve all `Humanoid` entities where the specified `outfit` parameter is a member of the `outfits` collection associated with each `Humanoid` entity.

When you execute this query, you need to set a value for the `:outfit` parameter to specify which outfit you are searching for. The result of the query will be a list of `Humanoid` entities that have the specified outfit in their `outfits` collection.

First Level Cache

https://vladmihalcea.com/jpa-hibernate-first-level-cache/

Problem with persist():

If you call the `persist` method on a detached entity, the JPA provider will throw an `IllegalArgumentException`.

The `persist` method is used to transition a new entity (which has never been associated with any persistence context) to a managed state. It is typically used for new entities to be persisted for the first time.

Here's what happens:

1. **Detached Entity**: A detached entity is an entity that was previously managed by a persistence context but is no longer associated with it. This could happen, for example, when the persistence context is closed or cleared.

2. **Persistence Context**: When you call `persist`, the JPA provider expects the entity to be in a transient (new) state, not in a detached state.

3. **IllegalArgumentException**: If you call `persist` on a detached entity, the JPA provider will throw an `IllegalArgumentException` to indicate that the operation is not allowed. This is because `persist` is not intended to be used with detached entities.

To persist a detached entity, you typically need to reattach it to a persistence context using methods like `merge` or `reattach` before calling `persist`. Alternatively, you can use `merge` directly to persist the detached entity by merging its state into a new or managed entity instance.

Problem with save():

In JPA, there's no `save` method defined in the EntityManager interface. However, if you are referring to the `save` method in Spring Data JPA's JpaRepository interface, the behavior depends on whether the entity is detached or not.

1. **Detached Entity**: If the entity is detached (i.e., not associated with a persistence context), calling `save` will treat it as a new entity and attempt to persist it. It will perform an `INSERT` operation in the database to save the entity as a new row.

2. **Managed Entity**: If the entity is managed (i.e., associated with a

persistence context), calling `save` is typically used to update the entity's state. Spring Data JPA's `save` method will determine whether the entity is new or existing based on its identifier. If the entity has a null identifier or does not exist in the database, it will perform an `INSERT` operation to save it as a new row. If the entity has a non-null identifier and exists in the database, it will perform an `UPDATE` operation to update its state.

It's important to note that the behavior may vary depending on the specific implementation of the `save` method and any custom logic implemented in your application. Always consult the documentation of the specific JPA provider or Spring Data JPA for detailed information on the behavior of the `save` method in your environment.

Brief:

Yes, that's correct. In the context of JPA (Java Persistence API), a detached entity refers to an entity object that was previously associated with a persistence context but is no longer actively managed by it.

Here's a breakdown:

- **Managed Entity**: When an entity is retrieved from the database or created using the `new` operator within the scope of a persistence context (typically within a transaction), it becomes a managed entity. The persistence context keeps track of changes to managed entities and automatically synchronizes those changes with the database upon transaction commit.

- **Detached Entity**: If a managed entity is removed from the persistence context, either explicitly by calling `EntityManager.detach(entity)` or implicitly when the persistence context is closed or cleared, it becomes detached. Detached entities still represent data in the database, but any changes made to them will not be automatically synchronized with the database. They are essentially disconnected from the persistence context.

Detached entities are often used in scenarios where you need to work with entities outside the scope of a persistence context, such as transferring data between layers of an application or passing them across network boundaries. However, when you want to make changes to a detached entity persistent, you'll need to reattach it to a persistence context before those changes can be synchronized with the database. This is typically done using methods like `merge` or by reassociating the entity with a new persistence context.