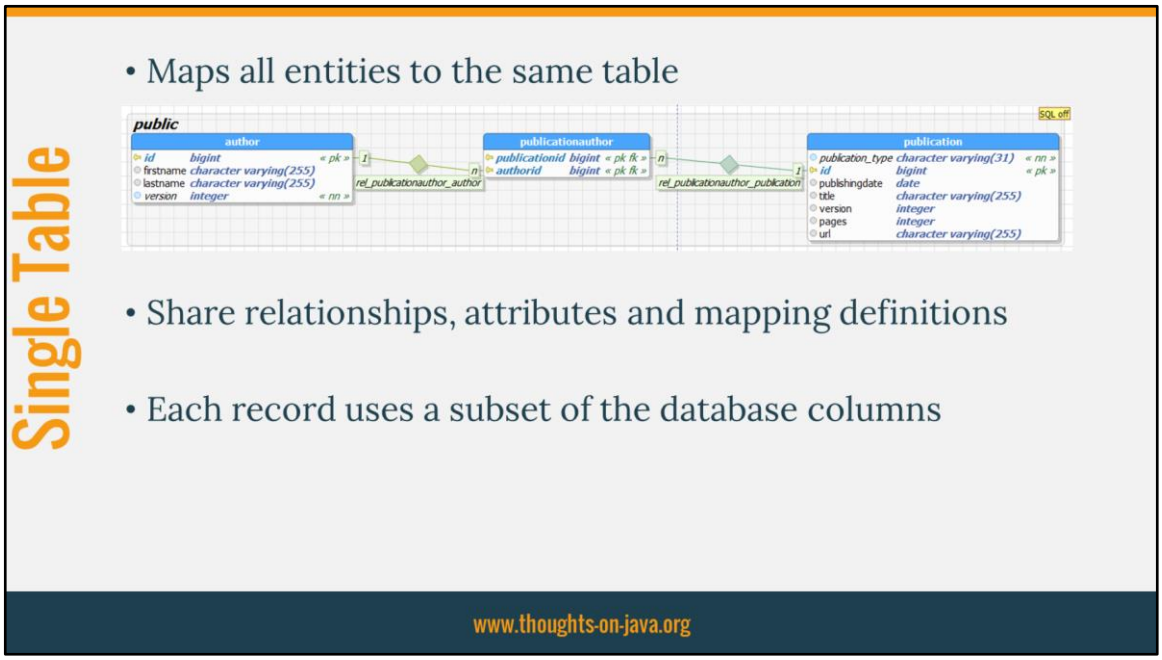
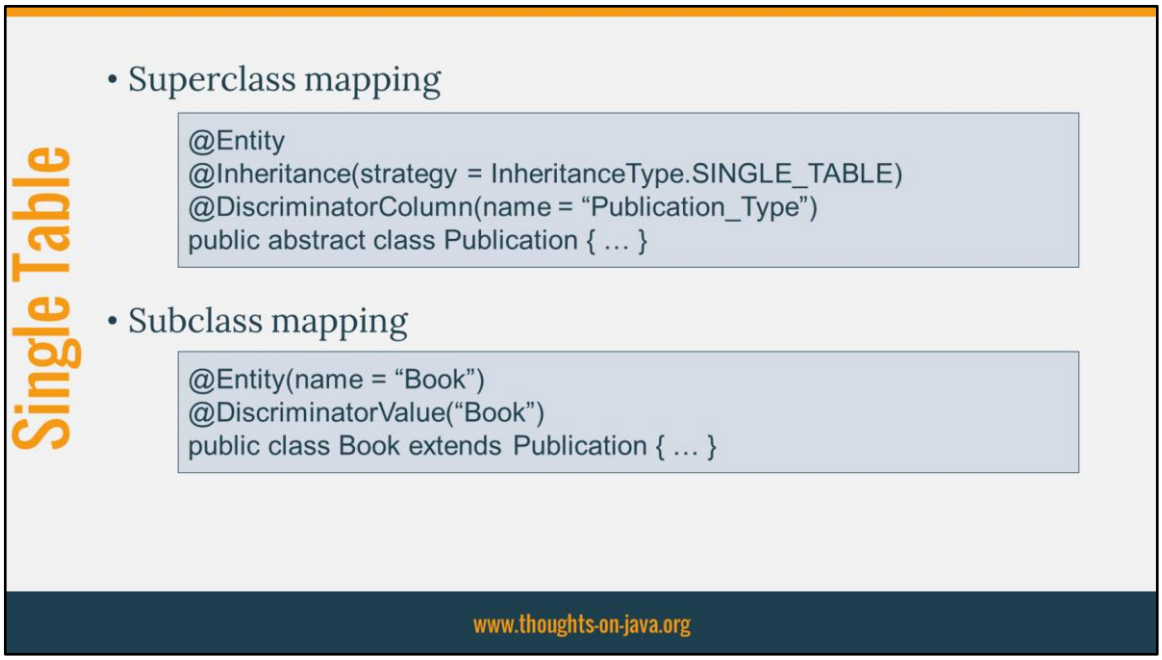
The single table strategy is the 3rd inheritance strategy I want to show you. It takes a different approach compared to the previously explained strategies to gain better performance. But as always, you don't get the benefits for free.



As you might have guessed from its name, the single table strategy maps all entities of the inheritance structure to the same database table. It allows you to share attributes and their mapping definitions and also to define polymorphic queries and relationships. In contrast to the previously discussed table per class strategy, the single table approach can do this with simple and fast SQL statements. That makes it the fastest inheritance strategy available.
But it also has a huge drawback. Each entity maps only a subset of the database columns and sets the rest of them to null. That makes it impossible to define not null constraints on any database columns that are not mapped by all entities in the hierarchy. It prevents you from ensuring data consistency on a database level, and you need to carefully decide if you want to take this risk.

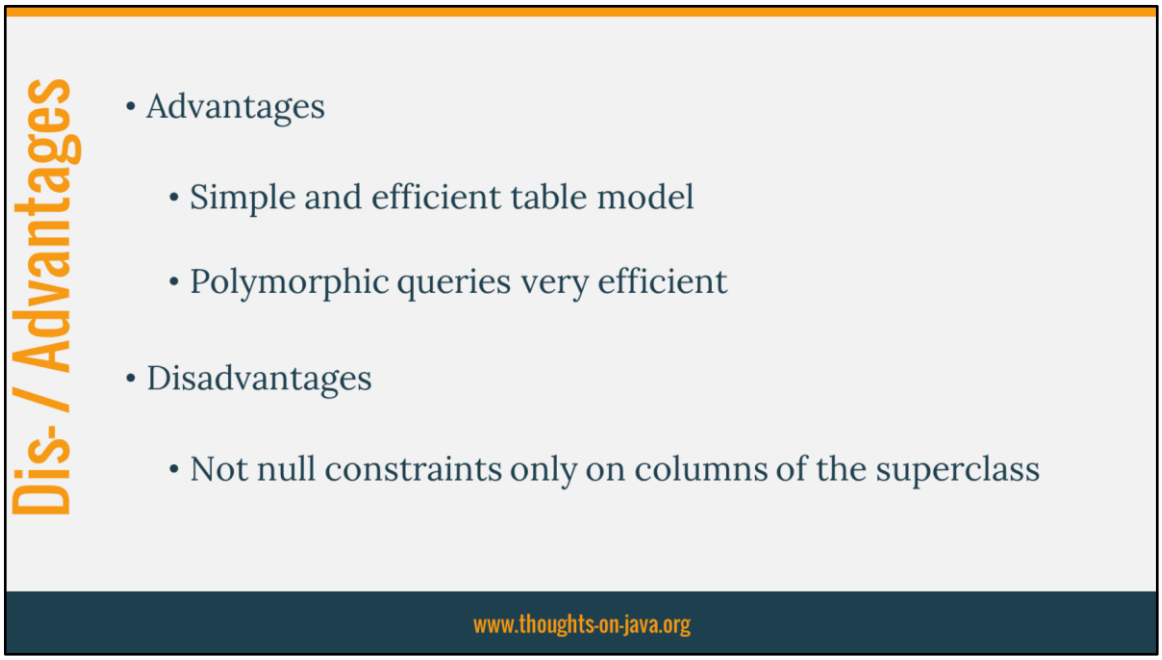


The mapping definition of the single table strategy requires more annotations than for the other ones.
As you can see in the first code snippet, the Publication class gets annotated with @Entity and @Inheritance. @Inheritance defines the inheritance strategy and sets it to InheritanceType.SINGLE_TABLE.
You can also provide a @DiscriminatorColumn annotation. When you persist multiple entities in the same database table, you need to store additional information to distinguish the different entity classes. That is done with a discriminator column that contains a different value for each entity class. The @DiscriminatorColumn annotation allows you to define the name of this column. If you don't provide it, all JPA implementations will use the default name DTYPE.

If you want to comply with the JPA standard, you also need to add a @DiscriminatorValue annotation to each subclass. It defines the discriminator value for this specific entity and is optional when you use Hibernate. If you don't provide it, Hibernate will use the simple entity name by default. But you might not be able to switch to a different JPA implementation without adapting your code.

Demo

www.thoughts-on-java.org



The two main advantages of the single table strategy are simplicity and efficiency. This strategy provides the best performance, especially for polymorphic queries and relationships.
But you don't get the performance advantages for free. Storing all entities in the same database table prevents you from using not null constraints on columns that are not mapped by all entities. You need to decide for your application and table model, if this increases the risk of data inconsistency and if you want to take it to get better performance.

Exercises

www.thoughts-on-java.org