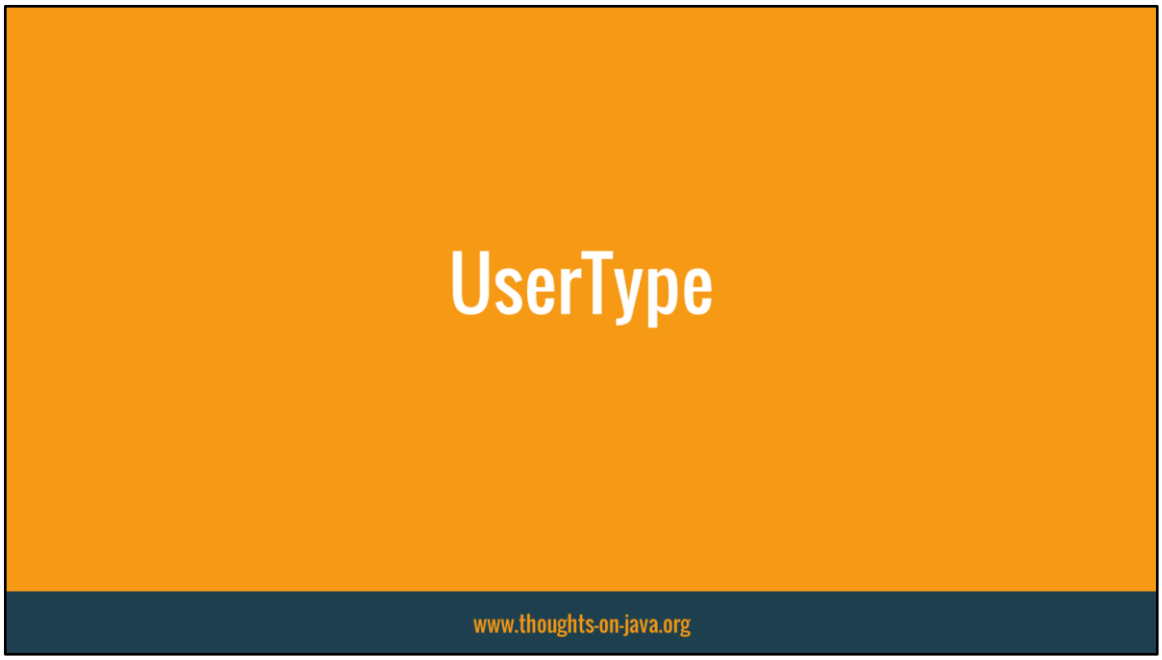
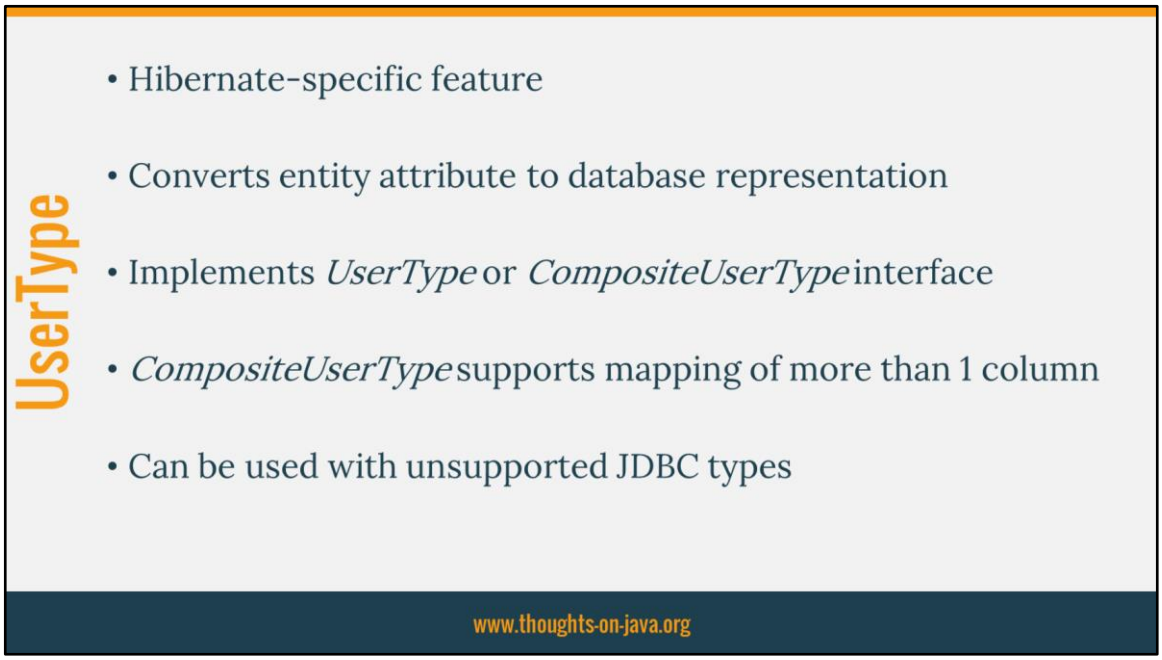


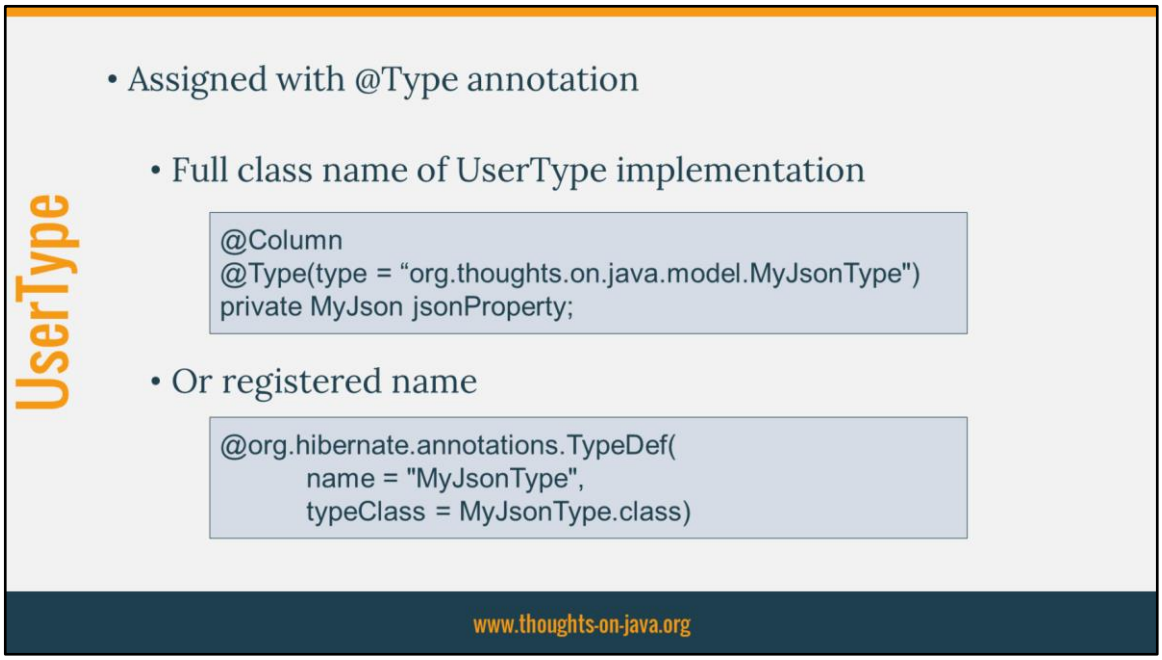
UserTypes are a Hibernate-specific feature to implement custom conversion of entity attribute types to database column types. They are more powerful than *AttributeConverter* but not portable to other JPA implements and not as easy to implement.



There are different interfaces a *UserType* can implement. You can find all of them in the *org.hibernate.usertype* package. The two most common ones are the *UserType* and the *CompositeUserType* interface. Both of them require you to implement several methods to convert the entity attribute value to its database representation and back.

In most cases, you will implement the *UserType* interface. It allows you to map an entity attribute to one database column. You could do the same with an *AttributeConverter*. The only advantage of the *UserType* is that you can also use it to map your entity attribute to an unsupported JDBC type. As I explained in the previous video, the Hibernate bug HHH-9553 prevents you from doing that with an *AttributeConverter*.

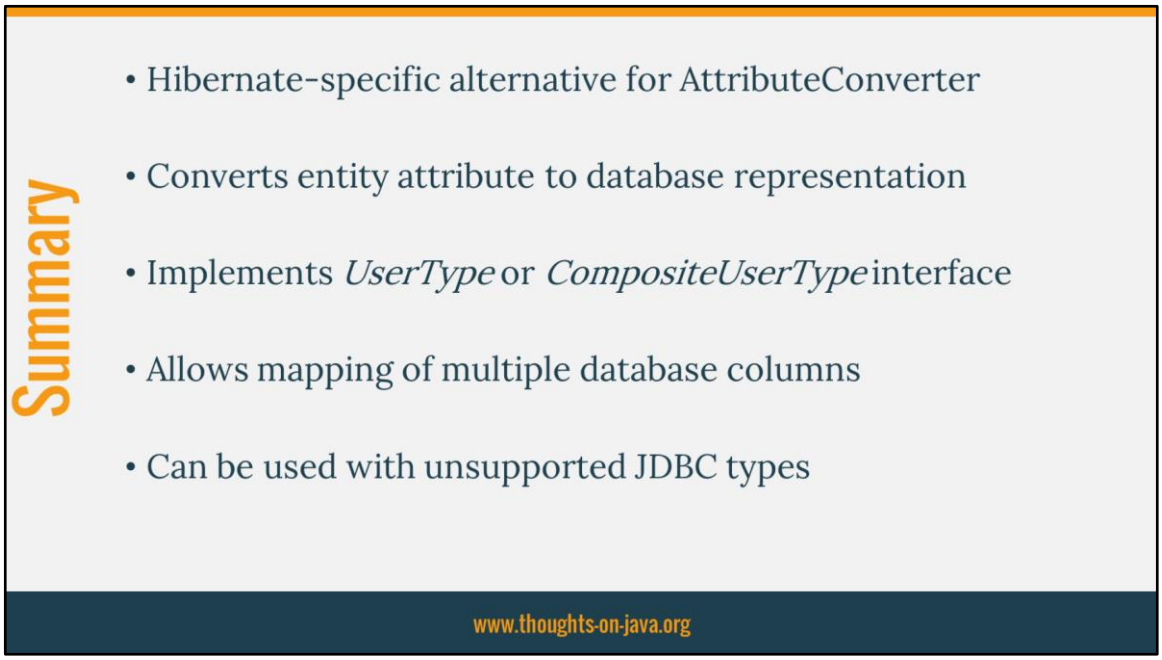
The *CompositeUserType* interface supports the mapping of an entity attribute to more than one database column. That is something you can't do with a JPA *AttributeConverter*, and you, therefore, need to use the more complex *UserType* approach.



When you've implemented your *UserType*, you need to assign it to an entity attribute. You can easily do that with Hibernate's *@Type* annotation. You can see an example of it here on the slide.
If you haven't registered the *UserType*, you have to provide the fully qualified class name to the type attribute of the *@Type* annotation. If you don't want to do that, you can register the *UserType* with a *@TypeDef* annotation, as I did in the second code snippet. You can then reference it by its name in the *@Type* annotation.
Let's switch to the IDE and have a more detailed look at it.

Demo

www.thoughts-on-java.org



As you've seen, *UserTypes* are a Hibernate-specific alternative for JPA's *AttributeConverter*. They implement one of the interfaces of the *org.hibernate.usertype* package. Most often it's either the *UserType* interface, if you want to map to only one database column, or the *CompositeUserType*, to map multiple database columns. The mapping of multiple database columns is also one of the advantages of Hibernate's *UserTypes*. Another one is the option to use them with unsupported JDBC types. I showed you an example of that in the IDE when we implemented a *UserType* that maps PostgreSQL's *JSONB* data type.

Exercises

www.thoughts-on-java.org