WAY2JAVA
a blog for Java

HOME TABLE OF CONTENTS

Differences between Table per class Table per subclass Table per concreteclass

HIBERNATE TUTORIALS

Differences between Table per class Table per subclass Table per concreteclass

S. Nageswara Rao, Corporate Trainer December 15, 2013

1 Comment

Table per class vs Table per subclass vs Table per concreteclass

To understand better, it is summarized all the differences between three hierarchical models of Hibernate.

1. **Table Per Class**
2. **Table Per Subclass**
3. **Table Per Concreteclass**

Introduction notes for the above three models is available at Inheritance (Hierarchical) Mapping Tutorial

	TABLE-PER-CLASS	TABLE-PER-SUBCLASS	TABLE-PER-CONCRETECLASS
NO. TABLES CREATED FOR CLASSES INVOLVED IN HIERARCHY	Only one table is created for all the classes	One table for each class is created	One table for each concrete class (subclass) is created but not of super classes
EXTRA FIELD/KEY	One Discriminator field is used	Foreign key is maintained between the tables	Foreign key is not maintained
SPECIAL FEATURE	Super class cannot be abstract or interface	Super class cannot be abstract or interface	Super class can be also an abstract or interface

Search:

Custom Search

OTHER POSTS IN SERIES

XML MAPPING ELEMENT	<subclass>	<joined-subclass>	<union-subclass>
POLYMORPHIC SUPPORT BETWEEN ENTITIES	Provides good support	Provides poor support	Provides poor support
NULL FIELDS	Some columns of subclasses will be null	Need not be null	Need not be null
ADVANTAGE	Gives maximum performance even for the classes involved in deep hierarchy	Changes in a single super class does not demand complex changes in database	Simple style to implement inheritance mapping
DISADVANTAGE	Requires either columns to be added or removed when the members in the hierarchy change	For deep hierarchy, it may give poor performance	Members of super class are placed in each subclass and thereby any change in the super class should be reflected in each subclass

Limitations:

1. Hibernate does not allow to write all the three elements <subclass>, <joined-subclass> and <union-subclass> mappings in one root <class> element.
2. It is allowed to place both the table-per-class and table-per-subclass models under one <class> element.
3. It is allowed to write all the three elements <subclass>, <joined-subclass> and <union-subclass> mappings in separate mapping files under one <hibernate-mapping>.
4. In table-per-class, columns declared by the subclasses should not have the constraint NOT NULL.

- limitations of Table per subclass Table per concrete class
- Table-per-class vs Table-per-concreteclass
- Table-per-class vs Table-per-subclass
- Table-per-subclass vs Table-per-concreteclass

← Table Per Concreteclass Example Hibernate

Relational Associations Tutorial Hibernate →

Hibernate Tutorial JDBC vs Hibernate
Create Database Connection Hibernate
Hibernate Simple Example Basics
Insert Record
Hibernate Simple Program Step-by-step Execution
hbm2ddl HibernateSessionFactory
Hibernate
Reverse Engineering Hibernate
Example
get(), load(), delete(), Update()
Hibernate
Difference between Eager Initialization
Lazy Initialization
Difference bewteen session.get
session.load Hibernate
Difference Query.list() Query.iterate()
Hibernate
list() iterate() Lazy Eager Initialization
Hibernate
Pagination Example Hibernate
First level Second level Cache Example
Hibernate
Inheritance Hierarchical Mapping
Tutorial Hibernate
Table Per Class Example Hibernate
Table Per Subclass Example Hibernate
Table Per Concreteclass Example
Hibernate
Differences between Table per class
Table per subclass Table per
concreteclass
Relational Associations Tutorial
Hibernate
One-to-One Example Hibernate
One-to-Many Example Hibernate
Many-to-One Example Hibernate
Many-to-Many Example Hibernate

LIKE US ON FACEBOOK

Java

Way2Java
15,275 likes

Like Page Conta

Be the first of your friends to like thi

One thought on “Differences between Table per class Table per subclass Table per concreteclass”

1. Rajeeev
December 21, 2015 at 2:18 pm

Great explanation…..the best way to understand Inheritance mapping.

Reply ↓

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

POST COMMENT

□ Notify me of follow-up comments by email.

□ Notify me of new posts by email.

Copyright © 2018 Way2Java