Parul® University

NAAC A++

Enterprise Programming using JAVA

Chapter-4: Hibernet (ORM)

Prof. ARNIKA PATEL

Assistant Professor

Department of CSE

Parul® University

NAAC A++

Content

INDEX

Dialects

- **Dialect** is a class that acts as a bridge between **Java JDBC types** and **SQL types**, which contains the mapping between java language data type and database datatype.

- Dialect allows Hibernate to generate SQL optimized for a particular relational database. Hibernate generates queries for the specific database based on the **Dialect** class.

- A hibernate dialect gives information to the framework of how to convert **hibernate queries(HQL)** into native **SQL queries**.

Dialects

Dialects can be used in the following ways:

- To generate Optimized SQL queries
- To interact with a particular Database if the application works with the help of more than one Database.
- To set default values for hibernate configuration file properties based on the database software even though they are not specified in the configuration file.

Dialects

SQL Dialects Configuration

The SQL dialect converts the **HQL query** which we write in java or any other object-oriented program to the specific database SQL query.

For connecting any hibernate application with the database, it is required to provide the configuration of SQL dialect.

Parul® University NAAC A++

Dialects

SQL Dialects Configuration

We can provide it in **hibernate.cfg.xml (DB2 Dialect)** as :

```
<hibernate-configuration>
        <session-factory name="session-factory">
                <property
name="hibernate.dialect">org.hibernate.dialect.DB2Dialect
</property>
        </session-factory>
</hibernate-configuration>
```

We can also specify in the properties file as :

hibernate.dialect=org.hibernate.dialect.DB2Dialect

Dialects

The **hibernate.dialect** property should be set to the correct org.hibernate.dialect.

Dialect subclass for the application database. If the Dialect class is not specified in the configuration, for most of the databases, Hibernate tries to resolve dialect names from the database connections.

But It is best to provide dialect so that Hibernate identifies the appropriate Dialect class for specific database versions.

Mapping

The hibernate works to link the JAVA language to the database table along with this link we can establish relations/mappings.

The main basic types of mapping are:

Primitive Types

Date and Time Types

Binary and Large Object Types

JDK-related Types

Mapping

Primitive Types

Mapping Type	Java Type	ANSI SQL Type
integer	int or java.lang.Integer	INTEGER
character	java.lang.String	CHAR(1)
float	float or java.lang.Float	FLOAT
string	java.lang.String	VARCHAR
double	double or java.lang.Double	DOUBLE
boolean	boolean or java.lang.Boolean	BIT
short	short or java.lang.Short	SMALLINT
long	long or java.lang.Long	BIGINT
byte	byte or java.lang.Byte	TINYINT
big_decimal	java.math.BigDecimal	NUMERIC

Mapping

Date and Time

Mapping type	Java type	ANSI SQL Type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
calendar	java.util.Calendar	TIMESTAMP
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar_date	java.util.Calendar	DATE

Mapping

Binary and large objects

Mapping type	Java type	ANSI SQL Type
clob	java.sql.Clob	CLOB
blob	java.sql.Blob	BLOB
binary	byte[]	VARBINARY (or BLOB)
text	java.lang.String	CLOB
serializable	any Java class that implements java.io.Serializable	VARBINARY (or BLOB)

Mapping

JDK linked

Mapping type	Java type	ANSI SQL Type
class	java.lang.Class	VARCHAR
locale	java.util.Locale	VARCHAR
currency	java.util.Currency	VARCHAR
timezone	java.util.Currency	VARCHAR

Annotations

Annotation in JAVA is used to represent supplemental information. As you have seen @override, @inherited, etc are an example of annotations in general Java language

The motive of using a hibernate is to skip the SQL part and focus on core java concepts.

Generally, in hibernate, we use XML mapping files for converting our POJO classes data to database data and vice-versa.

But using XML becomes a little confusing so, in replacement of using XML, we use annotations inside our POJO classes directly to declare the changes.

Annotations

Also using annotations inside out POJO classes makes things simple to remember and easy to use.

Annotation is a powerful method of providing metadata for the database tables and also it gives brief information about the database table structure and also POJO classes simultaneously.

Annotations

Annotations	Use of annotations
@Entity	Used for declaring any POJO class as an entity for a database
@Table	Used to change table details, some of the attributes are- • name – override the table name • schema • catalogue • enforce unique constraints
@Id	Used for declaring a primary key inside our POJO class
@GeneratedValue	Hibernate automatically generate the values with reference to the internal sequence and we don't need to set the values manually.

Annotations

Annotations	Use of annotations
@Column	It is used to specify column mappings. It means if in case we don't need the name of the column that we declare in POJO but we need to refer to that entity you can change the name for the database table. Some attributes are- • Name – We can change the name of the entity for the database • length – the size of the column mostly used in strings • unique – the column is marked for containing only unique values • nullable – The column values should not be null. It's marked as NOT
@Transient	Tells the hibernate, not to add this particular column
@Temporal	This annotation is used to format the date for storing in the database
@Lob	Used to tell hibernate that it's a large object and is not a simple object

PPT Content Resources Reference Sample:

1. **Book Reference**

   Jim Farley, William Crawford, David Flanagan. Java Enterprise in a Nutshell, O'Reilly

2. **Book Reference**

   Rocha, R., Purificação, J. (2018). Java EE 8 Design Patterns and Best Practices: Build Enterprise-ready Scalable Applications with Architectural Design Patterns. Germany: Packt Publishing..

3. **Website Reference**

   https://www.scribd.com/document/268349254/Java-8-Programming-Black-Book .

4. **Sources**

   https://developers.redhat.com/topics/enterprise-java

5. **Article**

   https://www.researchgate.net/publication/276412369_Advanced_Java_Programming

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/