Вопросы и ответы на собеседование по Hibernate

(46-60)

46). Как мне указать имя таблицы, связанной с объектом, используя аннотацию?

```
@Entity
@Table(name="users")
public class User{
String username;
String password;
}
```

Как видно из приведенного выше кода, аннотация **@Table** используется для указания имени таблицы базы данных, связанной с объектом. Для этого объекта требуется обязательное name атрибута, которое указывает имя таблицы, как в базе данных.

47). Как переменная в сущности соединяется со столбцом базы данных?

По умолчанию Hibernate ищет имена столбцов, соответствующие именам переменных в классе. Однако также возможно указывать разные имена переменных и связывать их с соответствующими столбцами в базе данных.

48). Как указать другое имя столбца для отображения переменных?

Аннотация **@Column** используется для определения имени столбца, связанного с переменной. В отсутствие этой аннотации Hibernate предварительно компилирует отображение переменной, сопоставленной со столбцом с тем же именем. Пример использования этой аннотации показан ниже:

```
@Entity
@Table(name="users")
public class User{
@Column(name="user_name")
String username;
String password;
}
```

Атрибут **name** является обязательным атрибутом для указания имени столбца, отличного от имени переменной. Из приведенного выше кода можно понять, что столбец **user_name** связан с переменной **username**

49). Как мы указываем переменную, которая будет первичным ключом для таблицы?

Hibernate может создавать таблицы базы данных для приложения непосредственно на основе отображений, представленных в коде Java. В таком случае Hibernate требует знать, какие столбцы должны быть первичными ключами. Это можно настроить с помощью аннотации **@Id** . Hibernate не только заботится о создании этого столбца как столбца первичного ключа, но также проверяет свое уникальное ограничение при каждой вставке и обновлении базы данных.

50). Как мы определяем логику генерации значения первичного ключа?

Значения первичного ключа могут быть сгенерированы различными способами в зависимости от базы данных. Например, в базе данных MySQL первичные ключи могут быть сгенерированы с использованием алгоритма автоинкрементации, в то время как в базе данных Oracle вам необходимо создать последовательность и использовать ее для автоматического увеличения значения для первичного ключа. Эти методы генерации могут быть указаны с помощью приведенного ниже кода аннотации.

```
@Entity
@Table(name="users")
public class User{

@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
int userid;

@Column(name="user_name")
String username;

String password;
}
```

Столбец идентификатора пользователя здесь определен как первичный ключ, автоматически сгенерированный с использованием стратегии идентификации. Возможные значения для strategy включают в себя:

- GenerationType.AUTO
- GenerationType.IDENTITY
- GenerationType.SEQUENCE
- GenerationType.TABLE

51). Как вы настраиваете диалект в hibernate.cfg.xml?

Конфигурация диалекта в xml включает определение свойства с именем hibernate.dialect. Пример XML-тега для определения диалекта показан ниже:

```
<property name="org.hibernate.dialect.MySQLDialect">
org.hibernate.dialect.MySQLDialect
</property>
```

52). Как настроить URL базы данных и учетные данные в hibernate.cfg.xml?

Hibernate Framework может быть настроен с использованием различных значений свойств. Пример URL базы данных конфигурации и учетных данных приведен ниже.

<property name = "hibernate.connection.url">jdbc:mysql://localhost/mydb</property>

<property name = "hibernate.connection.username">root</property>

<property name = "hibernate.connection.password">password</property>

При выполнении приложения Java платформа Hibernate предварительно компилирует код для подключения к базе данных и создает пул подключений, чтобы уменьшить накладные расходы на подключение к базе данных во время выполнения запросов.

53). Как настроить размер пула соединений?

Размер пула соединений в Hibernate имеет два значения — минимальный размер пула и максимальный размер пула. Эти размеры можно настроить с помощью свойств **hibernate.c3p0.min_size** и **hibernate.c3p0.max_size** . Эти свойства можно настроить так же, как показано выше для учетных данных базы данных.

54). Как мы совершаем транзакцию в Hibernate?

Объект транзакции в Hibernate может быть зафиксирован или откатан. Для выполнения этого действия мы используем приведенный ниже код.

```
tx = Session.beginTransaction();
...
...
...
//Do something with transaction
...
...
...
tx.commit();
```

Как видно, вызов функции **tx.commit()** выполняет задачу транзакции в базе данных. Для отката процедура остается прежней. Все, что вам нужно сделать, это изменить вызов функции на **tx.rollback()** .

55). Можно ли подключить несколько баз данных в одном приложении Java с помощью Hibernate?

Да. Практически можно подключить одно приложение Java к нескольким базам данных, используя два отдельных файла конфигурации Hibernate и две отдельные SessionFactory. Эти файлы конфигурации содержат различные конфигурации диалектов, относящихся к соответствующей базе данных. Объекты исключительно отображаются в соответствующую конфигурацию базы данных. Таким образом, с двумя разными параллельными объектами SessionFactory можно подключить несколько баз данных.

56). Поддерживает ли Hibernate полиморфизм?

Да. Hibernate по своей природе поддерживает полиморфизм. Hibernate классы в основном поддерживают свои операции посредством самого полиморфизма.

57). Сколько сессий Hibernate существует в любой момент времени в приложении?

Hibernate сессия является общим объектом. В любой момент времени существует только один общий объект сеанса, который помогает в управлении транзакциями и получении соединений из пула соединений. Следует отметить, что это верно только при использовании одной конфигурации базы данных. В случае нескольких конфигураций базы данных Hibernate создаст отдельный объект сеанса для поддержки сопоставления и транзакций для другой базы данных.

58). Какие изоляции транзакций есть в Hibernate?

Hibernate поддерживает четыре уровня изоляции транзакций, которые могут быть заданы с помощью аннотаций или XML-конфигурации:

- **READ_UNCOMMITTED** - это наименьший уровень изоляции, который позволяет одной транзакции видеть изменения, внесенные другой транзакцией до их фиксации. Этот уровень может привести к "грязному чтению", когда транзакция видит данные, которые могут быть отменены.
- **READ_COMMITTED** - это уровень изоляции по умолчанию в Hibernate. Он гарантирует, что транзакция видит только изменения, зафиксированные другими транзакциями. Это предотвращает "грязное чтение", но может привести к "неповторяемому чтению" при повторном чтении данных, которые были изменены другой транзакцией между двумя чтениями.
- **REPEATABLE_READ** - это уровень изоляции, который гарантирует, что транзакция видит одни и те же данные при повторном чтении в рамках той же самой транзакции. Транзакция не видит изменения, внесенные другими транзакциями после начала текущей транзакции.
- **SERIALIZABLE** - это наивысший уровень изоляции, который гарантирует, что транзакция видит данные в том же самом состоянии, что и при начале транзакции. Он предотвращает "грязное чтение", "неповторяемое чтение" и "фантомное чтение", но может привести к замедлению производительности.

Выбор уровня изоляции зависит от требований к приложению и конкретных сценариев использования.

59). Чем отличаются JPA и Hibernate?

В начале стоит отметить, что **JPA (Java Persistence API)** это **спецификация**, а Hibernate **это реализация** этой спецификации. Спецификация JPA предоставляет стандарты для ORM (Object-Relational Mapping) технологий в Java. Это означает, что JPA определяет набор правил и руководств по тому, как должен работать ORM-инструмент.

60). Как интегрировать Hibernate и Spring?

Лучше всего прочитать о настройках на сайтах фреймворков для текущей версии. Оба фреймворка поддерживают интеграцию из коробки и в общем настройка их взаимодействия не составляет труда. Общие шаги выглядят следующим образом.

- Добавить зависимости для hibernate-entitymanager, hibernate-core и spring-orm.
- Создать классы модели и передать реализации DAO операции над базой данных. Важно, что DAO классы используют SessionFactory, который внедряется в конфигурации бинов Spring.
- Настроить конфигурационный файл Spring (смотрите в офф. документации).
- Дополнительно появляется возможность использовать аннотацию @Transactional и перестать беспокоиться об управлении транзакцией Hibernate.