Вопросы и ответы на собеседование по Hibernate

(1-15)

1). Что такое Hibernate Framework?

Hibernate — библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения (**object-relational mapping** — **ORM**).
Она представляет собой свободное программное обеспечение с открытым исходным кодом (**open source**).
Данная библиотека предоставляет легкий в использовании каркас (фреймворк) для отображения **объектно-ориентированной модели данных** в традиционные **реляционные** базы данных.
Основной особенностью фреймворка (и самой полезной его частью) является то, что он представляет базу данных в форме **объекта**. Эти объекты могут быть написаны без особых знаний в SQL. Это отличная возможность, так как помогает разработчикам сэкономить много времени – это очень важно в сфере современного программирования.
Hibernate также имеет встроенное объектное отображение – это минимизирует число строк кода, необходимых для работы приложения.

2). Что такое ORM?

ORM является аббревиатурой для “**Object-related Mapping**” или “**Объектно-реляционного Отображения**”.
Это фундаментальная концепция платформы Hibernate, которая сопоставляет таблицы базы данных с объектами Java, а затем предоставляет различные **API** для выполнения различных типов операций над таблицами данных.

3). Какие важные преимущества дает использование Hibernate Framework?

Hibernate является одним из самых востребованных **ORM** фреймворков для Java. И вот почему:

1. Hibernate устраняет множество повторяющегося кода, который постоянно преследует разработчика при работе с JDBC. Скрывает от разработчика множество кода, необходимого для управления ресурсами и позволяет сосредоточиться на бизнес логике.
2. Hibernate поддерживает **XML** так же как и **JPA** аннотации, что позволяет сделать реализацию кода независимой.
3. Hibernate предоставляет собственный мощный язык запросов (**HQL**), который похож на SQL. Стоит отметить, что HQL **полностью объектно-ориентирован** и понимает такие принципы, как наследование, полиморфизм и ассоциации (связи).
4. Hibernate — широко распространенный open source проект. Благодаря этому доступны тысячи открытых статей, примеров, а так же документации по использованию фреймворка.
5. Hibernate легко интегрируется с другими Java EE фреймворками, например, **Spring Framework** поддерживает встроенную интеграцию с Hibernate.
6. Hibernate поддерживает **ленивую инициализацию** используя **proxy** объекты и выполняет запросы к базе данных только по необходимости.
7. Hibernate поддерживает разные **уровни cache**, а следовательно может повысить производительность.
8. Важно, что Hibernate может использовать чистый SQL, а значит поддерживает возможность **оптимизации** запросов и работы с любым сторонним БД и его фичами.

4). Какие преимущества Hibernate над JDBC?

Hibernate имеет ряд преимуществ перед JDBC API:

1. Hibernate удаляет множество повторяющегося кода из JDBC API, а следовательно его легче читать, писать и поддерживать.
2. Hibernate поддерживает наследование, ассоциации и коллекции, что не доступно в JDBC API.
3. Hibernate неявно использует управление **транзакциями**. Большинство запросов нельзя выполнить вне транзакции. При использовании JDBC API для управления транзакциями нужно явно использовать commit и rollback.
4. JDBC API throws SQLException, которое относится к проверяемым исключениям, а значит необходимо **постоянно** писать множество блоков try-catch. В большинстве случаев это не нужно для каждого вызова JDBC и используется для управления транзакциями. Hibernate оборачивает исключения JDBC через непроверяемые JDBCException или HibernateException, а значит нет необходимости проверять их в коде каждый раз. Встроенная поддержка управления транзакциями в Hibernate убирает блоки try-catch.
5. Hibernate Query Language (HQL) более объектно ориентированный и близкий к Java языку, чем SQL в JDBC.
6. Hibernate поддерживает кэширование, а запросы JDBC — нет, что может понизить производительность.
7. Конфигурация Hibernate позволяет использовать JDBC вроде соединения по типу JNDI DataSource для пула соединений. Это важная фича для энтерпрайз приложений, которая полностью отсутствует в JDBC API.
8. Hibernate поддерживает аннотации **JPA**, а значит код является переносимым на другие ORM фреймворки, реализующие стандарт, в то время как код JDBC сильно привязан к приложению.

5). Назовите некоторые важные интерфейсы Hibernate.

1. **SessionFactory (org.hibernate.SessionFactory)** — неизменяемый потокобезопасный объект с компилированным маппингом для одной базы данных. Необходимо инициализировать SessionFactory **всего один раз**. Экземпляр SessionFactory используется для получения объектов Session, которые используются для операций с базами данных.
2. **Session (org.hibernate.Session)** — однопоточный короткоживущий объект, который предоставляет связь между объектами приложения и базой данных. Он **оборачивает** JDBC java.sql.Connection и работает как фабрика для org.hibernate.Transaction. Разработчик должен открывать сессию по необходимости и закрывать ее сразу после использования. Экземпляр Session является интерфейсом между кодом в java приложении и hibernate framework и предоставляет методы для операций **CRUD**.
3. **Transaction (org.hibernate.Transaction)** — однопоточный короткоживущий объект, используемый для атомарных операций. Это абстракция приложения от основных JDBC или JTA транзакций. org.hibernate.Session может занимать несколько org.hibernate.Transaction в определенных случаях.

6). Что такое конфигурационный файл Hibernate?

Файл конфигурации Hibernate содержит в себе данные о базе данных и необходим для инициализации **SessionFactory**. В **.xml** файле необходимо указать вендора базы данных или JNDI ресурсы, а так же информацию об используемом диалекте, что поможет hibernate выбрать режим работы с конкретной базой данных.

7). Что такое Hibernate mapping file?

Файл отображения (**mapping file**) используется для связи entity бинов и колонок в таблице базы данных. В случаях, когда не используются аннотации JPA, файл отображения .xml может быть полезен (например при использовании сторонних библиотек).

8). Назовите некоторые важные аннотации, используемые для отображения в Hibernate.

Hibernate поддерживает как аннотации из JPA, так и свои собственные, которые находятся в пакете org.hibernate.annotations. Наиболее важные аннотации JPA и Hibernate:

1. **javax.persistence.Entity**: используется для указания класса как entity bean.
2. **javax.persistence.Table**: используется для определения имени таблицы из БД, которая будет отображаться на entity bean.
3. **javax.persistence.Access**: определяет тип доступа, поле или свойство. Поле — является значением по умолчанию и если нужно, чтобы hibernate использовал методы getter/setter, то их необходимо задать для нужного свойства.
4. **javax.persistence.Id**: определяет primary key в entity bean.
5. **javax.persistence.EmbeddedId**: используется для определения составного ключа в бине.
6. **javax.persistence.Column**: определяет имя колонки из таблицы в базе данных.
7. **javax.persistence.GeneratedValue**: задает стратегию создания основных ключей. Используется в сочетании с javax.persistence.GenerationType enum.
8. **javax.persistence.OneToOne**: задает связь один-к-одному между двумя сущностными бинами. Соответственно есть другие аннотации OneToMany, ManyToOne и ManyToMany.
9. **org.hibernate.annotations.Cascade**: определяет каскадную связь между двумя entity бинами. Используется в связке с org.hibernate.annotations.CascadeType.
10. **javax.persistence.PrimaryKeyJoinColumn**: определяет внешний ключ для свойства. Используется вместе с org.hibernate.annotations.GenericGenerator и org.hibernate.annotations.Parameter.

9). Что вы знаете о Hibernate SessionFactory и как его сконфигурировать?

SessionFactory является фабрикой классов и используется для получения объектов session.
Фабрика сессий инициализируется на основе конфигурации Hibernate (например, через файл **hibernate.cfg.xml**) и предоставляет сессии для работы с базой данных.
Обычно в приложении имеется **только один** экземпляр SessionFactory и потоки, обслуживающие клиентские запросы, получают экземпляры session с помощью объекта SessionFactory.
Внутреннее состояние SessionFactory неизменно (**immutable**) и включает в себя все метаданные об **Object Relational Mapping** и задается при создании SessionFactory.
SessionFactory также предоставляет методы для **получения метаданных** класса и статистики, вроде данных о **втором уровне кэша**, выполняемых запросах и т.д.

10). Является ли Hibernate SessionFactory потокобезопасным?

Т.к. объект SessionFactory **immutable** (неизменяемый), **то да**, он потокобезопасный. Множество потоков может обращаться к одному объекту одновременно.

11). Как получить Hibernate Session и что это такое?

Объект Hibernate **Session** является связью между кодом java приложения и hibernate. Это **основной интерфейс** для выполнения операций с базой данных. Жизненный цикл объекта session связан с началом и окончанием транзакции. Сессия открывается в начале транзакции и закрывается по её завершении. Этот объект предоставляет методы для **CRUD** (create, read, update, delete) операций для объекта **персистентности**. С помощью этого экземпляра можно выполнять HQL, SQL запросы и задавать критерии выборки.

12). Является ли Hibernate Session потокобезопасным?

Объект Hibernate Session **не является потокобезопасным**. Каждый поток должен иметь свой собственный объект Session и закрывать его по окончанию.

13). В чем разница между openSession и getCurrentSession?

Hibernate SessionFactory **getCurrentSession()** возвращает сессию, связанную с контекстом. Но для того, чтобы это работало, нам нужно настроить его в конфигурационном файле hibernate. Так как этот объект session связан с контекстом hibernate, то отпадает необходимость к его закрытию. Объект session закрывается вместе с закрытием SessionFactory.

<property name="hibernate.current_session_context_class">thread</property>

Метод Hibernate SessionFactory **openSession() всегда создает новую сессию**. Мы должны обязательно контролировать **закрытие** объекта session по завершению всех операций с базой данных. Для многопоточной среды необходимо создавать новый объект session для каждого запроса. Существует еще один метод **openStatelessSession()**, который возвращает session **без поддержки состояния**. Такой объект **не реализует** первый уровень кэширования **и не** взаимодействует с вторым уровнем. Сюда же можно отнести игнорирование коллекций и некоторых обработчиков событий. Такие объекты могут быть полезны при загрузке больших объемов данных без удержания большого кол-ва информации в кэше.

14). Какая разница между методами Hibernate Session get() и load()?

get() и **load**() - это два разных метода для получения объектов из базы данных в Hibernate, и они имеют некоторые важные различия:

1) **get**():
   - Если объект не найден в базе данных, get() возвращает null.
   - Вызывая get(), Hibernate выполняет запрос к базе данных сразу же, чтобы найти объект и загрузить его в память.
   - Этот метод может использоваться, когда вы хотите получить объект из базы данных, и если он не существует, вам необходимо получить null в результате.

```
MyEntity entity = (MyEntity) session.get(MyEntity.class, id);
```

2) **load**():
   - Если объект не найден в базе данных, load() не возвращает null. Вместо этого, он создает прокси-объект (ленивую загрузку) и возвращает его. Прокси-объект не содержит реальных данных, и загрузка данных из базы данных происходит только тогда, когда к объекту обращаются (ленивая загрузка). Если объект существует, load() вернет его.
   - Этот метод может использоваться, когда вы уверены, что объект существует, и вы хотите использовать ленивую загрузку для оптимизации производительности.

```
MyEntity entity = (MyEntity) session.load(MyEntity.class, id);
```

Разница в поведении между get() и load() заключается в том, что get() возвращает null, если объект не найден, в то время как load() возвращает прокси-объект и выполняет ленивую загрузку. Выбор между этими методами зависит от требований вашего приложения и того, как вы хотите обрабатывать отсутствующие объекты.Нужно использовать метод get(), если необходимо удостовериться в наличии данных в БД.

15). Что вы знаете о кэшировании в Hibernate? Объясните понятие кэш первого уровня в Hibernate?

Hibernate использует кэширование, чтобы сделать наше приложение **быстрее**. Кэш Hibernate может быть очень полезным в получении высокой производительности приложения при правильном использовании.
Идея кэширования заключается в сокращении количества запросов к базе данных.
Кэш **первого уровня** Hibernate **связан с объектом Session и включен по умолчанию** и не существует никакого способа, чтобы его отключить. Однако Hibernate предоставляет методы, с помощью которых мы можем **удалить** выбранные объекты **из кэша** или полностью **очистить кэш**.
Любой объект закэшированный в session **не будет виден** другим объектам session. После закрытия объекта сессии все кэшированные объекты будут потеряны.