++ 2016 Hibernate | Вячеслав Круглов — Введение в Hibernate: что, зачем, и где стандартные ловушки

https://www.youtube.com/watch?v=C-wEZjEOhWc&t=1430s&ab_channel=JUG.ru

JDBC драйверы для каждой конкретной базы данных позволяют общаться с бд, отправлять запросы и т.д.

JPA - это спецификация, она предоставляет платформу объектно-реляционного отображения для управления данными между объектами / классами Java и реляционной базой данных в Java-приложениях.

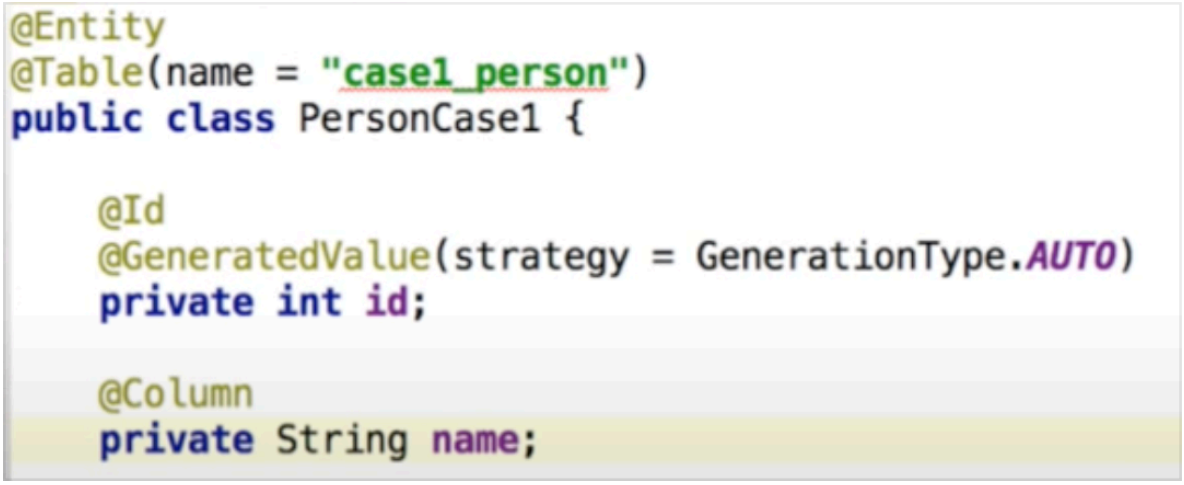
JPQL - это стандартный язык запросов сущностей JPA, в то время как **HQL** расширяет **JPQL** и добавляет некоторые специфические для **Hibernate** функции.

Hibernate - это наиболее широко используемая ORM-реализация руководящих принципов JPA.

HQL - The Hibernate Query Language. По сравнению с SQL. HQL полностью объектно-ориентирован и понимает такие понятия, как наследование, полиморфизм и ассоциация.
(from Cat as cat left join cat.mate.kittens as kittens)

Для вывода sql нужно в конфигурациях указать showSql = true

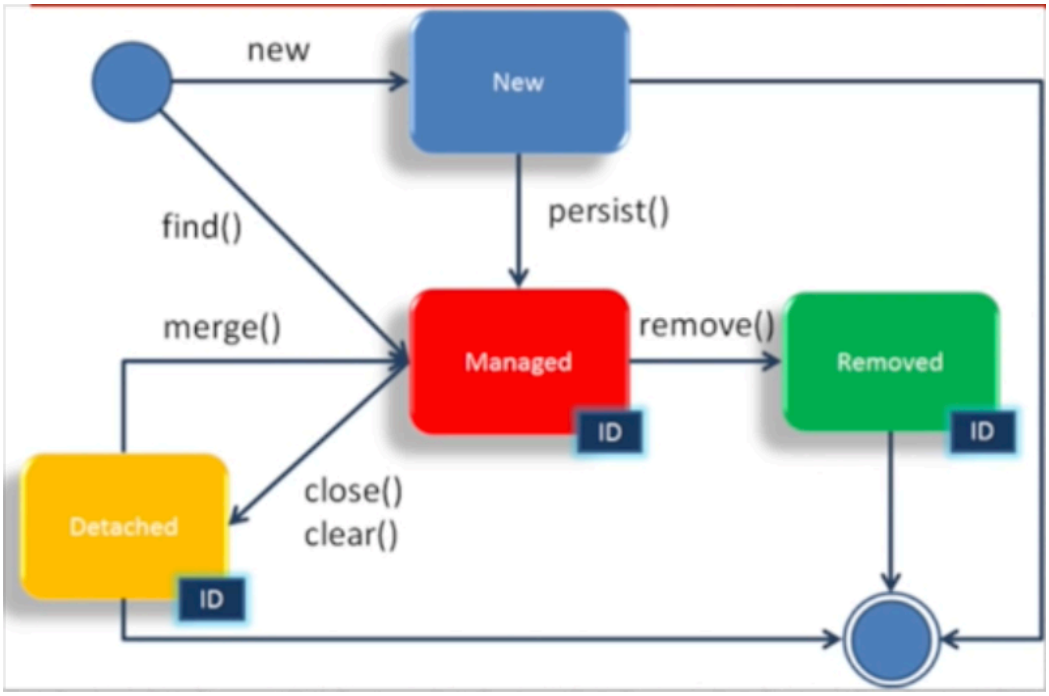
@Entity - простой клаас/сущность замапленная на таблицу базы данных



Entity Manager - интерфейс для доступа к базе данных

Percistance Context - набор "Entity", управляемый Entity Manager

Этапы жизненного цикла сущности



Когда сущность переходит в состояние Detached, она перестает управляться EntityManager. Отсюда можнт возникнуть LazyInitalisationException.

Если выполнить методы Clear до попадения сущностей в базу данных, то они просто потеряются.

Связи сущностей

- One-to-one
- One-to-many (many-to-one)
- Many-to-many

- Связи между сущностями могут быть настроены через аннотации

```
@OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
@JoinColumn(name = "person_id", updatable = false)
private Set<PetCase1> pets;
```

Hibernate имеет большое количество параметров установленных по умолчанию и нужно понимать как оно работает

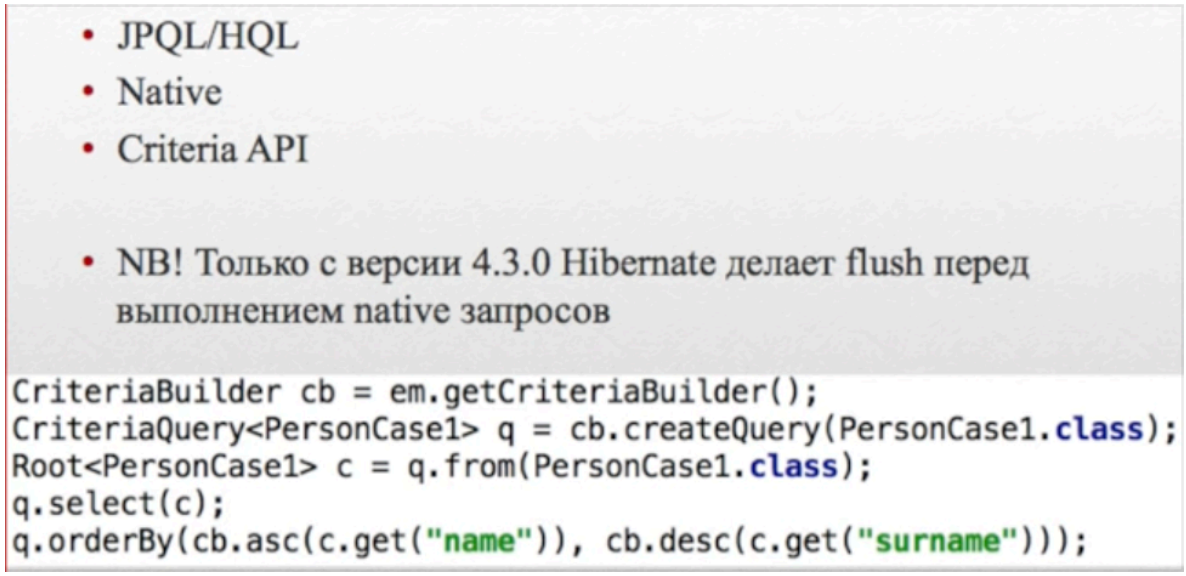
Flush policy

- Flush – выполнение накопленных в persistence context запросов в БД
- Существует несколько вариантов для настройки момента выполнения flush
- JPA – Commit и Auto

Выполнение запросов может быть отложено и их порядок может изменяться, например запрос может выполняться в конце транзакции.

ПРОВЕРЯТЬ По умолчанию в hibernate flush происходит перед выполнением select запроса для обеспечения консистентности данных

Существует три способа создания запросов



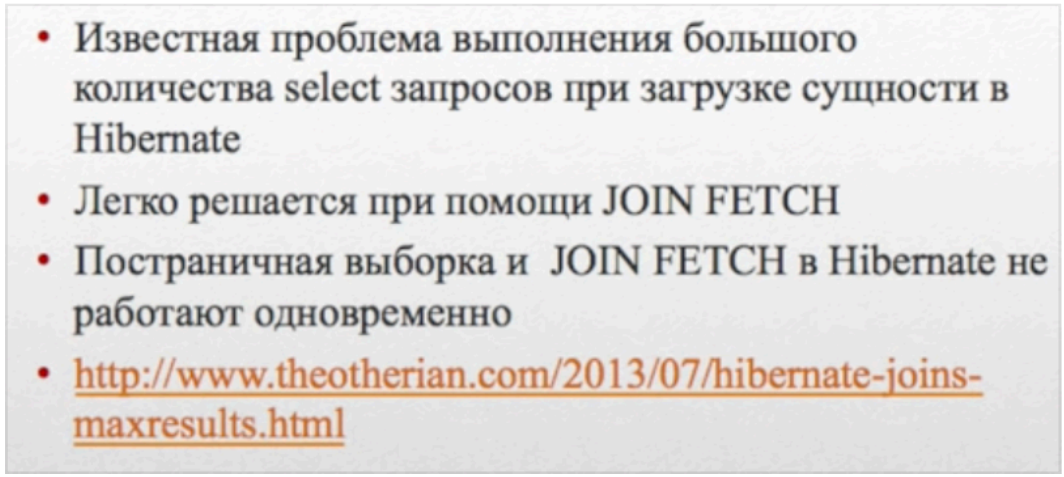
При отказе от нативных sql запросов переход между базами данных вполняется легче.

(1) С версии 4.3.0 Hybernate выполняет flush перед выполнение native Query.

До версии 4.3.0 при выполнении в одной транзакции nativeQuery и JPQL запросов, hybernate не сбрасывал изменения объектов из EntityManager в БД и native запрос прочитывал другие результаты.

После версии 4.3.0 выполниться два метода flush.

Проблемы Hibernate



Проблема N+1 запроса

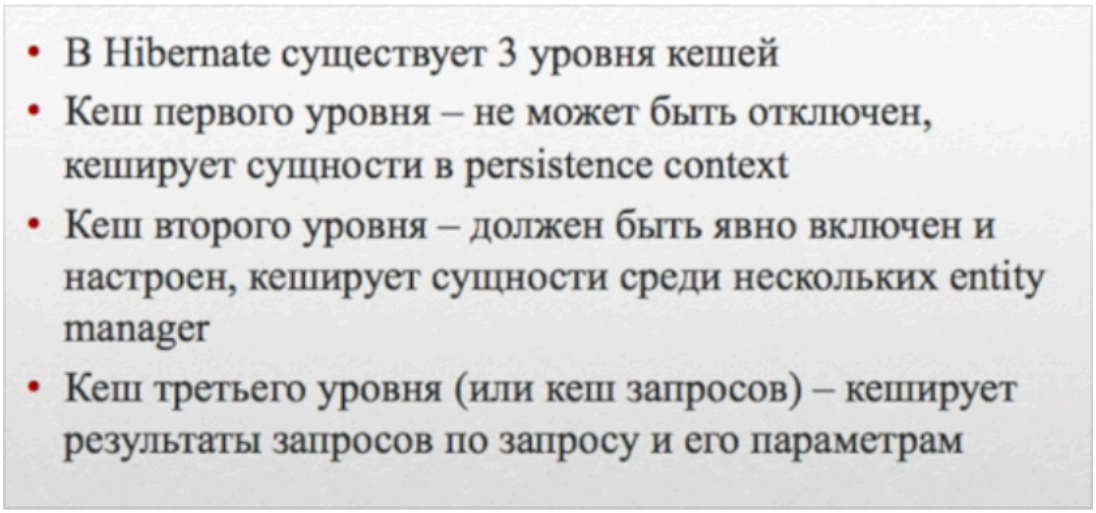
При большом количестве связей в сущностях выполнится большое количество запросов на получение дочерних сущностей. Это снижает производительность.

Добавляя JOIN FETCH в HQL будет выполнен только один запрос.

ПРОВЕРЯТЬ Однако есть проблема при ограничении количества результататов.
Добавление setMaxResults(10) или пагинации сработает вместе с JOIN FETCH не очевидно. Hibernate выполнить ограничение не силами базы данных, а получит все записи из бд, правратит в сущности и по сущностям произведет отбор.

Один из способов решения проблемы, сделать 2 селекта. В первом выбрать родительскую сущность, а во втором из базы добавть все что нужно.

Кеши в Hibernate



Дополнение к картинке.
1) Включен по умолчанияю. Кеш первого уровня, если сущность есть в persistance context, то hibernate не пойдет в бд.
2) Кеш второго уровня, необходимо подключать какую-либо реализацию в конфигурации. Самый популярный был ehcache.
3) Кэш 3 уровня кэширует не объекты, а их id, сами объекты получаются из кэша второго уровня.

Проблемы:
При изменении сущностей вне entityManager, в рамках другой транзакции или напрямую в БД кеш первого уровня не будет обновлен и будет виден старый результат. Для избегания этой проблемы нужно вызывать у EntityManager метод refrash.

Нормализация это хорошо, но когда нужно выжать производительность на максимум сущности могут объединяться. Это значительно быстрее, чем в запросе добавится еще один JOIN. Можно схлопнуть две таблицы в базе данных в одну, но при этом

оставить два разных объекта.

> Настраиваются при помощи аннотаций @Embedded и @Embeddable
>
> Полезны при денормализации схемы БД

Hibernate может сам писать логи, но гораздо удобнее использвоать Log4JDBC

При миграции между базами данных необходимо обращать внимания на баги, имебщиеся в драйвере той, на которую будет миграции и на костыли в работе с текущей.