Hibernate

SessionFactory

**SessionFactory** – это класс который потокобезопасный (и неизменяемый) представляет сопоставления модели предметной области приложения с базой данных.

**Методы:**

**close()** - уничтожьте это SessionFactory и освободите все его ресурсы, включая кэши и пулы соединений.

**containsFetchProfileDefinition​(String name)** - определите, существует ли определение профиля выборки, зарегистрированное под данным именем.

**findEntityGraphByName​(String name)** - верните корневой файл EntityGraph с заданным именем или null, если нет графика с заданным именем.

**findEntityGraphsByType​(Class<T> entityClass)** - возвращает все EntityGraph файлы, зарегистрированные для данного типа объекта.

**fromSession​(Function<Session,​R> action)** - откройте Session и используйте его для получения значения.

**fromTransaction​(Function<Session,​R> action)** - откройте Session и используйте его для выполнения действия в рамках транзакции.

**getCache()** - получите прямой доступ к базовым областям кэша.

**getCriteriaBuilder()** - получите, HibernateCriteriaBuilder который может быть использован для построения критериев.

**getCurrentSession()** - получает текущий сеанс, экземпляр Session, неявно связанный с некоторым контекстом или областью.

**getDefinedFetchProfileNames()** - получите набор имен всех defined fetch profiles.

**getDefinedFilterNames()** - получите набор имен всех defined filters.

**getSchemaManager()** - SchemaManager с тем же каталогом и схемой по умолчанию, что и объединенные соединения, принадлежащие этой фабрике.

**getStatistics()** - извлеките статистику для этой фабрики.

**inSession​(Consumer<Session> action)** - откройте Session и используйте его для выполнения действия.

**inTransaction​(Consumer<Session> action)** - откройте Session и используйте его для выполнения действия в рамках транзакции.

**isClosed()** - эта фабрика уже закрыта?

**openSession()** - откройте Session.

**openStatelessSession()** - откройте новый сеанс без состояния.

**openStatelessSession​(Connection connection)** - откройте новый сеанс без состояния, используя указанный JDBC Connection.

**withOptions()** - получите конструктор сеансов для создания новых Session файлов с определенными настраиваемыми параметрами.

**withStatelessOptions()** - получите StatelessSession конструктор.

Session

**Session** – это однопоточный недолговечный объект, концептуально моделирующий "Единицу работы" (PoEAA). За кулисами Hibernate Session оборачивает JDBC и действует как фабрика для экземпляров.

**Методы:**

**addEventListeners​(SessionEventListener... listeners)** - добавьте в сеанс одного или нескольких прослушивателей

**byId​(Class<T> entityClass)** - создайте IdentifierLoadAccess экземпляр для извлечения указанной сущности по первичному ключу.

**byId​(String entityName)** - создайте IdentifierLoadAccess экземпляр для извлечения указанного типа сущности по первичному ключу.

**byMultipleIds​(Class<T> entityClass)** - создайте MultiIdentifierLoadAccess экземпляр для одновременного извлечения нескольких объектов, как указано значениями первичного ключа.

**byMultipleIds​(String entityName)** - создайте MultiIdentifierLoadAccess экземпляр для одновременного извлечения нескольких объектов, как указано значениями первичного ключа.

**byMultipleNaturalId​(Class<T> entityClass)** - доступ для загрузки нескольких объектов с помощью natural id.

**byMultipleNaturalId​(String entityName)** - доступ для загрузки нескольких объектов с помощью natural id.

**byNaturalId​(Class<T> entityClass)** - создайте NaturalIdLoadAccess экземпляр для извлечения указанной сущности по ее естественному идентификатору.

**byNaturalId​(String entityName)** - создайте NaturalIdLoadAccess экземпляр для извлечения указанной сущности по ее естественному идентификатору.

**bySimpleNaturalId​(Class<T> entityClass)** - создайте SimpleNaturalIdLoadAccess экземпляр для извлечения указанной сущности по ее простому (единственному атрибуту) natural id.

**bySimpleNaturalId​(String entityName)** - создайте SimpleNaturalIdLoadAccess экземпляр для извлечения указанной сущности с помощью ее natural id.

**cancelQuery()** - отмените выполнение текущего запроса.

**clear()** - полностью очистите сеанс.

**contains​(String entityName, Object object)** - определите, связан ли данный объект с этим сеансом.

**createNamedQuery​(String name, Class<R> resultClass)** - создайте типизированный Query экземпляр для данного именованного запроса.

**createQuery​(jakarta.persistence.criteria.CriteriaQuery<R> criteriaQuery)** - создайте Query для данного JPA CriteriaQuery.

**createQuery​(String queryString, Class<R> resultClass)** - создайте типизированный Query экземпляр для заданной строки запроса HQL.

**detach​(Object object)** - удалите этот экземпляр из кэша сеанса.

**disableFetchProfile​(String name)** - отключите fetch profile с указанным именем в этом сеансе.

**disableFilter​(String filterName)** - отключите именованный фильтр для текущего сеанса.

**enableFetchProfile​(String name)** - включите fetch profile с указанным именем в этом сеансе.

**enableFilter​(String filterName)** - включите именованный фильтр для этого текущего сеанса.

**evict​(Object object)** - удалите этот экземпляр из кэша сеанса.

**flush()** - принудительно сбросьте этот сеанс.

**get​(Class<T> entityType, Object id)** - возвращает постоянный экземпляр данного класса сущностей с заданным идентификатором или null, если такого постоянного экземпляра нет.

**get​(Class<T> entityType, Object id, LockMode lockMode)** - возвращает постоянный экземпляр данного класса сущностей с заданным идентификатором или null, если такого постоянного экземпляра нет.

**get​(Class<T> entityType, Object id, LockOptions lockOptions)** - возвращает постоянный экземпляр данного класса сущностей с заданным идентификатором или null, если такого постоянного экземпляра нет.

**get​(String entityName, Object id)** - возвращает постоянный экземпляр заданной именованной сущности с заданным идентификатором или null, если такого постоянного экземпляра нет.

**get​(String entityName, Object id, LockMode lockMode)** - возвращает постоянный экземпляр данного класса сущностей с заданным идентификатором или null, если такого постоянного экземпляра нет.

**get​(String entityName, Object id, LockOptions lockOptions)** - возвращает постоянный экземпляр данного класса сущностей с заданным идентификатором или null, если такого постоянного экземпляра нет.

**getCacheMode()** - получите текущий cache mode для этого сеанса.

**getCacheRetrieveMode()** - определенный JPA CacheRetrieveMode.

**getCacheStoreMode()** - определенный JPA CacheStoreMode.

**getCurrentLockMode​(Object object)** - определите текущий LockMode данного управляемого экземпляра, связанного с этим сеансом.

**getEnabledFilter​(String filterName)** - извлеките включенный в данный момент фильтр по имени.

**getEntityName​(Object object)** - возвращает имя объекта для постоянного объекта.

**getFlushMode()** - получите текущий JPA flush mode для этого сеанса.

**getHibernateFlushMode()** - получите текущий flush mode для этого сеанса.

**getIdentifier​(Object object)** - возвращает значение идентификатора данного объекта, связанного с этим сеансом.

**getLobHelper()** - получите фабрику для экземпляров Blob и Clob.

**getReference​(Class<T> entityType, Object id)** - возвращает ссылку на постоянный экземпляр с заданным классом и идентификатором, предполагая, что экземпляр все еще является постоянным в базе данных.

**getReference​(String entityName, Object id)** - возвращает ссылку на постоянный экземпляр заданной именованной сущности с заданным идентификатором, предполагая, что экземпляр все еще является постоянным в базе данных.

**getReference​(T object)** - возвращает ссылку на постоянный экземпляр с тем же идентификатором, что и данный экземпляр, который может быть отсоединен, исходя из предположения, что экземпляр все еще является постоянным в базе данных.

**getSessionFactory()** - получите фабрику сеансов, которая создала этот сеанс.

**getStatistics()** - получите статистику для этого сеанса.

**isDefaultReadOnly()** - будут ли объекты и прокси, загруженные в этот сеанс, по умолчанию доступны только для чтения?

**isDirty()** - содержит ли этот сеанс какие-либо изменения, которые должны быть синхронизированы с базой данных? Другими словами, будут ли выполняться какие-либо операции DML, если мы сбросим этот сеанс?

**isFetchProfileEnabled​(String name)** - включен ли fetch profile с указанным именем в этом сеансе?

**isReadOnly​(Object entityOrProxy)** - доступен ли указанный объект или прокси-сервер только для чтения?

**load​(Object object, Object id)** - считывает постоянное состояние, связанное с данным идентификатором, в данный переходный экземпляр.

**lock​(Object object, LockMode lockMode)** - получите указанный уровень блокировки для данного управляемого экземпляра, связанного с этим сеансом.

**lock​(Object object, LockOptions lockOptions)** - получите блокировку для данного управляемого экземпляра, связанного с этим сеансом, используя данный lock options.

**merge​(String entityName, T object)** - скопируйте состояние данного объекта в постоянный объект с тем же идентификатором.

**merge​(T object)** - скопируйте состояние данного объекта в постоянный объект с тем же идентификатором.

**persist​(Object object)** - сделайте временный экземпляр постоянным и пометьте его для последующей вставки в базу данных.

**persist​(String entityName, Object object)** - сделайте временный экземпляр постоянным и пометьте его для последующей вставки в базу данных.

**refresh​(Object object)** - перечитайте состояние данного управляемого экземпляра, связанного с этим сеансом, из базовой базы данных.

**refresh​(Object object, LockMode lockMode)** - перечитайте состояние данного управляемого экземпляра из базовой базы данных, получив заданное LockMode.

**refresh​(Object object, LockOptions lockOptions)** - перечитайте состояние данного управляемого экземпляра из базовой базы данных, получив заданное LockMode.

**remove​(Object object)** - отметьте экземпляр сохранения, связанный с этим сеансом, для удаления из базовой базы данных.

**sessionWithOptions()** - получите Session конструктор с возможностью копирования определенной информации из этого сеанса.

**setCacheMode​(CacheMode cacheMode)** - установите текущий cache mode для этого сеанса.

**setCacheRetrieveMode​(jakarta.persistence.CacheRetrieveMode cacheRetrieveMode)** - включить или отключить чтение из кэша второго уровня.

**setCacheStoreMode​(jakarta.persistence.CacheStoreMode cacheStoreMode)** - включить или отключить запись в кэш второго уровня.

**setDefaultReadOnly​(boolean readOnly)** - измените значение по умолчанию для объектов и прокси, загруженных в этот сеанс, с изменяемого на режим только для чтения или с изменяемого на режим только для чтения.

**setFlushMode​(jakarta.persistence.FlushModeType flushMode)** - установите текущий JPA flush mode для этого сеанса.

**setHibernateFlushMode​(FlushMode flushMode)** - установите текущий flush mode для этого сеанса.

**setReadOnly​(Object entityOrProxy, boolean readOnly)** - установите неизменяемый постоянный объект в режим только для чтения или объект только для чтения в изменяемый режим.

Transaction

**Transaction** – это однопоточный недолговечный объект, используемый приложением для разграничения границ отдельных физических транзакций.

**Методы:**

**getStatus()** - получите текущий статус этой транзакции.

**getTimeout()** - извлеките время ожидания транзакции, установленное для этого экземпляра.

**markRollbackOnly()** - попытайтесь пометить базовую транзакцию только для отката.

**registerSynchronization​(jakarta.transaction.Synchronization synchronization)** - зарегистрируйте пользователя synchronization callback для этой транзакции.

**setTimeout​(int seconds)** - установите время ожидания транзакции для любой транзакции, запущенной любым последующим вызовом EntityTransaction.begin() в этом экземпляре.

 Типы отображения

**Примитивные типы Java -** boolean, int, double и т.д.

**Примитивные оболочки -** Boolean, Integer, Double и т.д.

**Строки -** String

**Числовые типы произвольной точности -** BigInteger и BigDecimal

**Типы даты / времени -** LocalDate, LocalTime, LocalDateTime, OffsetTime, OffsetDateTime, Instant

**Устаревшие типы даты и времени -** Date и Calendar

**Устаревшие типы даты и времени -** Date, Time, Timestamp

**Массивы байтов и символов -** byte[] или Byte[], char[] или Character[]

**Перечислимые типы Java -** Любой enum

**Сериализуемые типы -** Любой тип, который реализует java.io.Serializable[1]

**@Entity(name = "") –** указывает накакую таблицу ссылается класс.

**@Id** – указывает что переменная это id.

**@Basic –** указывает что это базовый тип данных, необязательна так-как предполагается по умолчанию.

**optional** - логическое значение (по умолчанию равно true) Определяет, допускает ли этот атрибут значения null.

**fetch** - FetchType (по умолчанию используется значение EAGLEED) Определяет, следует ли извлекать этот атрибут быстро или лениво. EAGER указывает, что значение будет выбрано как часть загрузки владельца. LAZY значения извлекаются только при обращении к значению.

**@Column** - определяет правила для неявного определения имен таблиц и столбцов.

**@Formula** - позволяет отображать любое вычисленное значение базы данных в виде виртуального столбца, доступного только для чтения.

**@Enumerated -** первоначальный способ отображения перечислений.

**ORDINAL -** сохраняется в соответствии с порядковым номером значения enum в классе enum, как указано java.lang.Enum#ordinal.

**STRING -** сохраняется в соответствии с именем перечисляемого значения, как указано java.lang.Enum#name.

**@Size** – позволяет указать размер строки.

**@JdbcTypeCode** - в качестве альтернативы вы можете явно указать тип.

**@JavaType** - указывает один из стандартных маппингов.

**@Nationalized** – сопоставляет строки как национализированные.

**@Lob –** указывает что это большой объект.

**MutabilityPlan** это средство, с помощью которого Hibernate понимает, как обращаться со значением домена с точки зрения его внутренней изменчивости, а также связанных с этим проблем, таких как создание копий.

**На MutabilityPlan отображение может влиять любая из следующих аннотаций:**

* @Mutability
* @Immutable
* @MapKeyMutability
* @CollectionIdMutability

**Hibernate проверяет следующие места на @Mutability и @Immutable в порядке приоритета:**

1. Локально для сопоставления
2. О соответствующем AttributeConverter классе реализации (если таковой имеется)
3. О типе Java значения

**@Type** – указывается для пользовательского сопоставления типов.

**@TimeZoneStorage –** указывает на хранилище информации о часовом поясе.

**@GeneratedValue** – указывает на то, что значение само генерируется.

**@Convert –** указывает что данные преобразуются, допустим в класс.