Hibernate шпаргалка

⌘Khasang

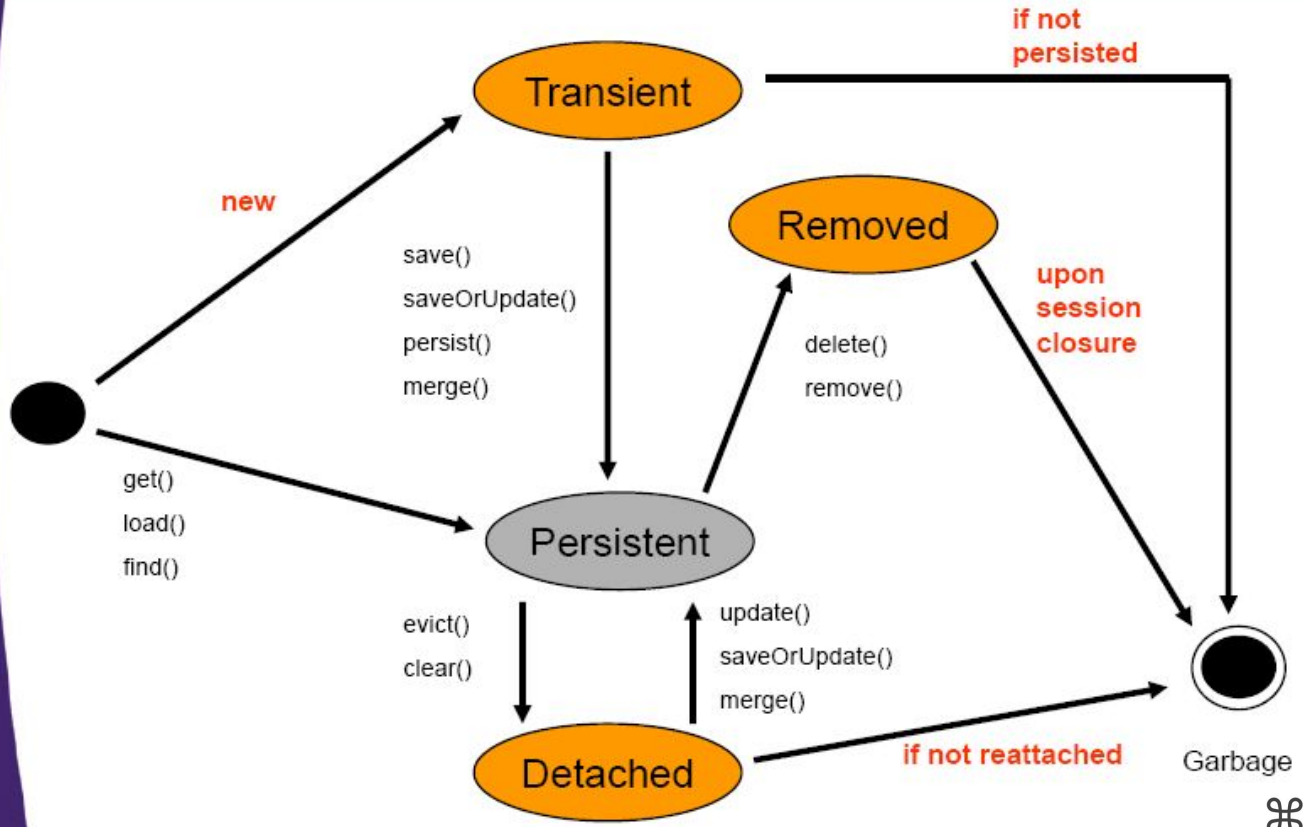
Hibernate FAQ. Де-факто корпоративный стандарт.

Целью Hibernate является освобождение разработчика от значительного объёма сравнительно низкоуровневого программирования по обеспечению хранения объектов в реляционной базе данных. Разработчик может использовать Hibernate как в процессе проектирования системы классов и таблиц «с нуля», так и для работы с уже существующей базой данных.

Hibernate не только решает задачу связи классов Java с таблицами базы данных (и типов данных Java с типами данных SQL), но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC-кода. Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки результирующего набора данных и преобразования объектов, максимально облегчая перенос (портирование) приложения на любые базы данных SQL.

Hibernate обеспечивает прозрачную поддержку сохранности данных (persistence) для «POJO» (то есть для стандартных Java-объектов); единственное строгое требование для сохраняемого класса — наличие конструктора по умолчанию (без параметров). Для корректного поведения в некоторых приложениях требуется также уделить внимание методам *equals()* и *hashCode(). Источник - https://ru.wikipedia.org/wiki/Hibernate*

Hibernate Lifecycle



Типичные запросы Criteria

Selecting an entity

```
CriteriaBuilder builder = entityManager.getCriteriaBuilder();

CriteriaQuery<Person> criteria = builder.createQuery( Person.class );
Root<Person> root = criteria.from( Person.class );
criteria.select( root );
criteria.where( builder.equal( root.get( Person_.name ), "John Doe" ) );

List<Person> persons = entityManager.createQuery( criteria ).getResultList();
```

Типичные запросы Criteria

Selecting an expression

```
CriteriaBuilder builder = entityManager.getCriteriaBuilder();

CriteriaQuery<String> criteria = builder.createQuery( String.class );
Root<Person> root = criteria.from( Person.class );
criteria.select( root.get( Person_.nickName ) );
criteria.where( builder.equal( root.get( Person_.name ), "John Doe" ) );

List<String> nickNames = entityManager.createQuery( criteria ).getResultList();
```

Типичные запросы Criteria

Selecting multiple values

```
CriteriaBuilder builder = entityManager.getCriteriaBuilder();

CriteriaQuery<Object[]> criteria = builder.createQuery( Object[].class );
Root<Person> root = criteria.from( Person.class );

Path<Long> idPath = root.get( Person_.id );
Path<String> nickNamePath = root.get( Person_.nickName);

criteria.select( builder.array( idPath, nickNamePath ) );
criteria.where( builder.equal( root.get( Person_.name ), "John Doe" ) );

List<Object[]> idAndNickNames = entityManager.createQuery( criteria ).getResultList();
```

Hibernate type (org.hibernate.type package)	JDBC type	Java type	BasicTypeRegistry key(s)
StringType	VARCHAR	java.lang.String	string, java.lang.String
MaterializedClob	CLOB	java.lang.String	materialized_clob
TextType	LONGVARCHAR	java.lang.String	text
CharacterType	CHAR	char, java.lang.Character	char, java.lang.Character
BooleanType	BIT	boolean, java.lang.Boolean	boolean, java.lang.Boolean
NumericBooleanType	INTEGER, 0 is false, 1 is true	boolean, java.lang.Boolean	numeric_boolean
YesNoType	CHAR, 'N'/'n' is false, 'Y'/'y' is true. The uppercase value is written to the database.	boolean, java.lang.Boolean	yes_no

TrueFalseType	CHAR, 'F'/'f' is false, 'T'/'t' is true. The uppercase value is written to the database.	boolean, java.lang.Boolean	true_false
ByteType	TINYINT	byte, java.lang.Byte	byte, java.lang.Byte
ShortType	SMALLINT	short, java.lang.Short	short, java.lang.Short
IntegerTypes	INTEGER	int, java.lang.Integer	int, java.lang.Integer
LongType	BIGINT	long, java.lang.Long	long, java.lang.Long
FloatType	FLOAT	float, java.lang.Float	float, java.lang.Float
DoubleType	DOUBLE	double, java.lang.Double	double, java.lang.Double
BigIntegerType	NUMERIC	java.math.BigInteger	big_integer, java.math.BigInteger
BigDecimalType	NUMERIC	java.math.BigDecimal	big_decimal, java.math.bigDecimal

CalendarType	TIMESTAMP	java.util.Calendar	calendar, java.util.Calendar
CalendarDateType	DATE	java.util.Calendar	calendar_date
CalendarTimeType	TIME	java.util.Calendar	calendar_time
CurrencyType	java.util.Currency	VARCHAR	currency, java.util.Currency
LocaleType	VARCHAR	java.util.Locale	locale, java.utility.locale
TimeZoneType	VARCHAR, using the TimeZone ID	java.util.TimeZone	timezone, java.util.TimeZone
UrlType	VARCHAR	java.net.URL	url, java.net.URL
ClassType	VARCHAR (class FQN)	java.lang.Class	class, java.lang.Class
BlobType	BLOB	java.sql.Blob	blog, java.sql.Blob
ClobType	CLOB	java.sql.Clob	clob, java.sql.Clob
BinaryType	VARBINARY	byte[]	binary, byte[]

MaterializedBlobType	BLOB	byte[]	materized_blob
ImageType	LONGVARBINARY	byte[]	image
WrapperBinaryType	VARBINARY	java.lang.Byte[]	wrapper-binary, Byte[], java.lang.Byte[]
CharArrayType	VARCHAR	char[]	characters, char[]
CharacterArrayType	VARCHAR	java.lang.Character[]	wrapper-characters, Character[], java.lang.Character[]
UUIDBinaryType	BINARY	java.util.UUID	uuid-binary, java.util.UUID
UUIDCharType	CHAR, can also read VARCHAR	java.util.UUID	uuid-char
PostgresUUIDType	PostgreSQL UUID, through Types#OTHER, which complies to the PostgreSQL JDBC driver definition	java.util.UUID	pg-uuid

SerializableType	VARBINARY	implementors of java.lang.Serializable	Unlike the other value types, multiple instances of this type are registered. It is registered once under java.io.Serializable, and registered under the specific java.io.Serializable implementation class names.
StringNVarcharType	NVARCHAR	java.lang.String	nstring
NTextType	LONGNVARCHAR	java.lang.String	ntext
NClobType	NCLOB	java.sql.NClob	nclob, java.sql.NClob
MaterializedNClobType	NCLOB	java.lang.String	materialized_nclob
PrimitiveCharacterArrayNClobType	NCHAR	char[]	N/A
CharacterNCharType	NCHAR	java.lang.Character	ncharacter
CharacterArrayNClobType	NCLOB	java.lang.Character[]	N/A

Пройдите мощный курс по SQL и Hibernate

http://khasang.io/p/hibernate

⌘Khasang