what is new in Hibernate 6

Serhii Petrenko

Jakarta Persistence 3.x

javax.persistence.* ====> jakarta.persistence.*

RELIEF:

[Using IntelliJ IDEA's migration tool - JetBrains Guide]

Attributes mapping update

Hibernate 6

```
@Entity
@Table(name = "foos")
public class Foo {

    //Some attributes omitted
    @Column(name = "recipe")
    @JdbcTypeCode(SqlTypes.JSON)
    private Bar bar;
}
```

Hibernate 5

```
@Entity
@Table(name = "foos")
public class Foo {

   //Some attributes omitted
   @Column(name = "bar")
   @Type(type = "com.vladmihalcea.hibernate.type.json.JsonType")
   private Bar bar;
}
```

Better date storage

`@TimeZoneStorage`

By default, Hibernate uses the native RDBMS data type to store the date with its timezone. If such data type is not supported, Hibernate creates an additional column with the `_tz` suffix.

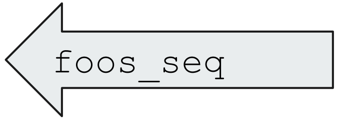
```
@Entity
@Table(name = "users")
public class User {
    @Column(name = "birth_date")
    @TimeZoneStorage
    private ZonedDateTime birthDate;
}
```

- Normalize the date according to the DB server timezone
- Normalize the date to UTC timezone
- Force Hibernate to use a column to store timezone

Sequence naming

- SingleNamingStrategy (Hibernate version < 5.3)
- LegacyNamingStrategy
- **StandardNamingStrategy** (Hibernate version >= 6.0)

```
@Entity
@Table(name = "foos")
public class Foo {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Integer id;
}
```



specify the sequence naming strategy explicitly
hibernate.id.db_structure_naming_strategy=single/legacy/standard

New UUID generation approach

```
@Id

@UuidGenerator(style = UuidGenerator.Style.TIME)

@GeneratedValue

private UUID id;
```

- Auto/Random – default strategy that uses the `UUID.randomUUID()` method.
- Time – time-based generation strategy consistent with IETF RFC 4122. Uses an IP address rather than a mac address. This strategy may be slower than the previous one.

Change in composite IDs

```java
@Entity

public class Book {

    @EmbeddedId

    private BookId id;

    private String genre;

    private Integer price;

}
```

Hibernate 5

```java
@Embeddable
public class BookId implements Serializable {

    private String author;
    private String name;
}
```

Hibernate 6

```java
@Embeddable
public class BookId {
    private String author;
    private String name;
}
```

Long text store improvements

up to 32K text stored

```
@JdbcTypeCode(SqlTypes.LONGVARCHAR)
@Column(name = "description")
private String description;
```

It will give us about 1Gb of text

```
@Column(name = "doc_txt", length = Length.LOB_DEFAULT)
private String docText;
```

still need more?

use the `@Lob` annotation

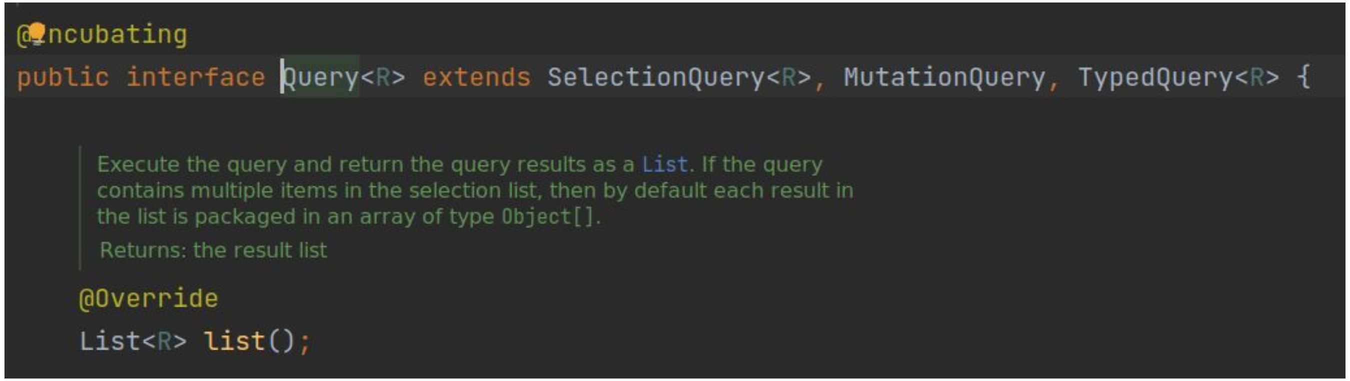
Better multitenancy

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

    @TenantId
    @Column(name = "tenant")
    private String tenant;
}
```

```java
@Component
public class TenantResolver implements
        CurrentTenantIdentifierResolver,
        HibernatePropertiesCustomizer {


    }
```

@Incubating annotation

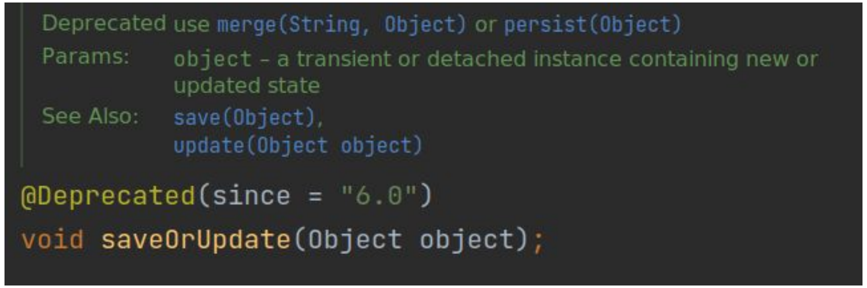
Marks the annotated Java element as incubating, potentially recursively. An incubating API is one which is still being actively developed and therefore may change at a later time; a "tech preview".



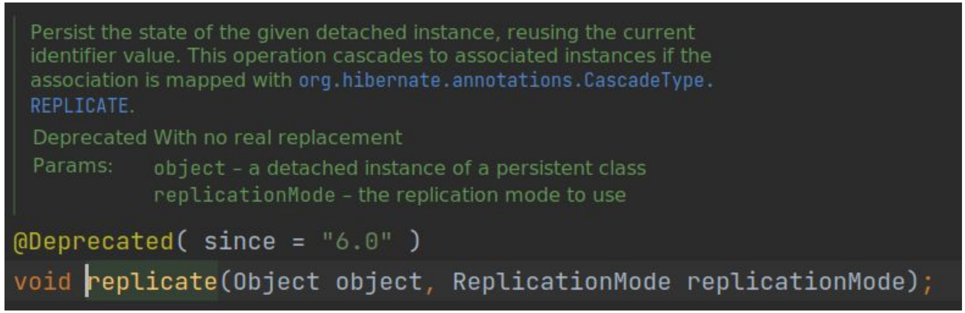
```
@Incubating
public interface Query<R> extends SelectionQuery<R>, MutationQuery, TypedQuery<R> {

    Execute the query and return the query results as a List. If the query
    contains multiple items in the selection list, then by default each result in
    the list is packaged in an array of type Object[].
    Returns: the result list
    @Override
    List<R> list();
```

Not so gorgeous

- EntityManager from JPA does not have saveOrUpdate
- a lot of methods from Session were marked @Deprecated



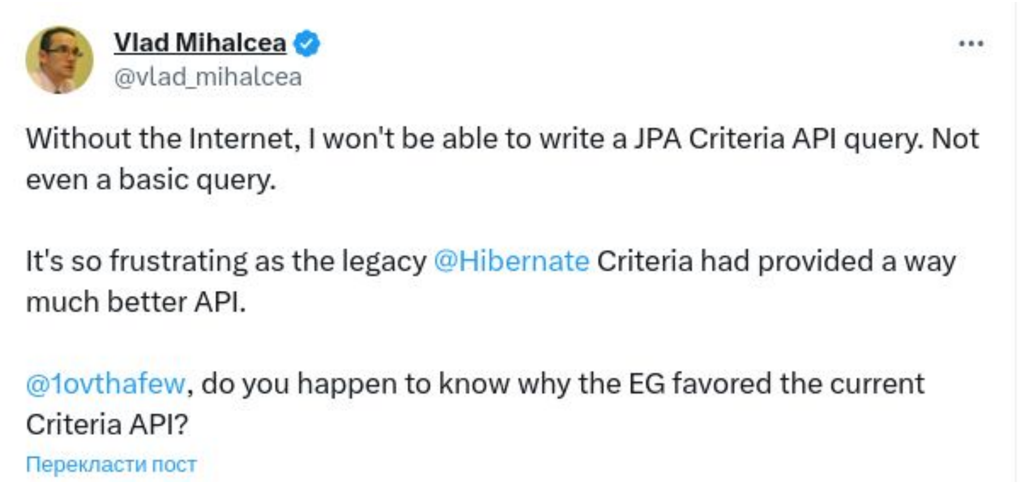
```
Deprecated use merge(String, Object) or persist(Object)
Params:    object – a transient or detached instance containing new or
           updated state
See Also:  save(Object),
           update(Object object)
@Deprecated(since = "6.0")
void saveOrUpdate(Object object);
```



```
Persist the state of the given detached instance, reusing the current
identifier value. This operation cascades to associated instances if the
association is mapped with org.hibernate.annotations.CascadeType.
REPLICATE.
Deprecated With no real replacement
Params:    object – a detached instance of a persistent class
           replicationMode – the replication mode to use
@Deprecated( since = "6.0" )
void replicate(Object object, ReplicationMode replicationMode);
```

Not so gorgeous

Goodbye, Hibernate Criteria!



Vlad Mihalcea
@vlad_mihalcea

Without the Internet, I won't be able to write a JPA Criteria API query. Not even a basic query.

It's so frustrating as the legacy @Hibernate Criteria had provided a way much better API.

@1ovthafew, do you happen to know why the EG favored the current Criteria API?

Перекласти пост

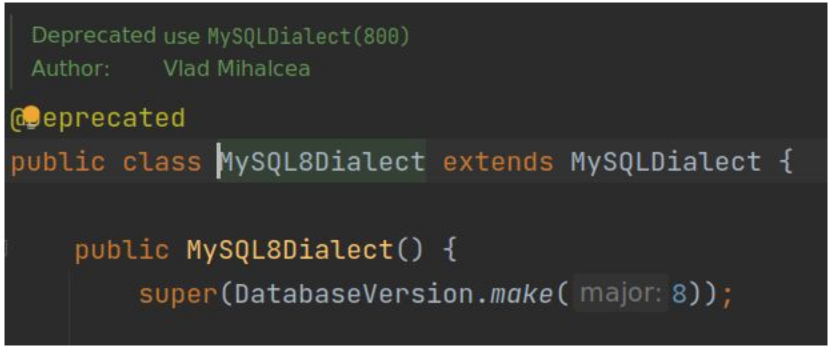
WHY?
https://discourse.hibernate.org/t/deprecation-of-hibernate-criteria-and-how-we-can-still-prevent-it/788/28

```
public List<Station> findAllByCriteria(StationCriteria stationCriteria) {
              return query(session -> {
                     Criteria criteria = session.createCriteria(Station.class);

                     if (stationCriteria.transportType() != null) {
                           criteria.add(Restrictions.eq(Station.FIELD_TRANSPORT_TYPE, stationCriteria.transportType()));
                     }

                     if (!StringUtils.isEmpty(stationCriteria.name())) {
                           criteria = criteria.createCriteria(Station.FIELD_CITY);
                           criteria.add(Restrictions.eq(City.FIELD_NAME, stationCriteria.name()));
                     }
                      return criteria.list();
               });
}
```

```
public List<Station> findAllByCriteria(StationCriteria stationCriteria) {
              return query(session -> {

                     CriteriaBuilder builder = session.getCriteriaBuilder();
                     CriteriaQuery<Station> criteria = builder.createQuery(Station.class);
                     Root<Station> root = criteria.from(Station.class);

                     List<Predicate> predicates = new ArrayList<>();
                     if (stationCriteria.transportType() != null) {
                           predicates.add(builder.equal(root.get(Station.FIELD_TRANSPORT_TYPE),
stationCriteria.transportType()));
                     }
                     if (!StringUtils.isEmpty(stationCriteria.name())) {
                         Join<Station, City> city = root.join(Station.FIELD_CITY);
                           predicates.add(builder.equal(city.get(City.FIELD_NAME), stationCriteria.name()));
                     }
                     Predicate predicate = builder.and(predicates.toArray(new Predicate[] {}));
                     criteria.select(root).where(predicate);
                     TypedQuery<Station> query = session.createQuery(criteria);
                     return query.getResultList();
              });
       }
```

Not so gorgeous

SchemaExport and
package org.hibernate.tool.hbm2ddl
were moved to *hibernate-ant*



```
Deprecated use MySQLDialect(800)
Author:    Vlad Mihalcea

@Deprecated
public class MySQL8Dialect extends MySQLDialect {

    public MySQL8Dialect() {
        super(DatabaseVersion.make( major: 8));
```

Useful links

Documentation - 6.3 - Hibernate ORM

Using IntelliJ IDEA's migration tool - JetBrains Guide

Migrating from ANTLR2 to ANTLR4 - Strumenta

The best way to use JOIN FETCH and Pagination with Spring

SnapAdmin