# JPA Annotations

1. @Access

2. @AssociationOverride

3. @AssociationOverrides

4. @AttributeOverride

5. @AttributeOverrides

6. @Basic

7. @Cacheable

8. @CollectionTable

9. @Column

10. @ColumnResult

11. @ConstructorResult

12. @Convert

13. @Converter

14. @Converts

15. @DiscriminatorColumn

16. @DiscriminatorValue

17. @ElementCollection

18. @Embeddable

19. @Embedded

20. @EmbeddedId

21. @Entity

22. @EntityListeners

23. @EntityResult

24. @Enumerated

25. @ExcludeDefaultListeners

26. @ExcludeSuperclassListeners

27. @FieldResult

28. @ForeignKey

29. @GeneratedValue

30. @Id

31. @IdClass

32. @Index

33. @Inheritance

34. @JoinColumn

35. @JoinColumns

36. @JoinTable

37. @Lob

38. @ManyToMany

39. @ManyToOne

40. @MapKey

41. @MapKeyClass

42. @MapKeyColumn

43. @MapKeyEnumerated

44. @MapKeyJoinColumn

45. @MapKeyJoinColumns

46. @MapKeyTemporal

47. @MappedSuperclass

48. @MapsId

49. @NamedAttributeNode

50. @NamedEntityGraph

51. @NamedEntityGraphs

52. @NamedNativeQueries

53. @NamedNativeQuery

54. @NamedQueries

55. @NamedQuery

56. @NamedStoredProcedureQueries

57. @NamedStoredProcedureQuery

58. @NamedSubgraph

59. @OneToMany

60. @OneToOne

61. @OrderBy

62. @OrderColumn

63. @PersistenceContext

64. @PersistenceContexts

65. @PersistenceProperty

66. @PersistenceUnit

67. @PersistenceUnits

68. @PostLoad

69. @PostPersist

70. @PostRemove

71. @PostUpdate

72. @PrePersist

73. @PreRemove

74. @PreUpdate

75. @PrimaryKeyJoinColumn

76. @PrimaryKeyJoinColumns

77. @QueryHint

78. @SecondaryTable

79. @SecondaryTables

80. @SequenceGenerator

81. @SqlResultSetMapping

82. @SqlResultSetMappings

83. @StoredProcedureParameter

84. @Table

85. @TableGenerator

86. @Temporal

87. @Transient

88. @UniqueConstraint

89. @Version

Let's describe each annotation with links to their Java doc and their official documentation sections.

# @Access

The @Access annotation is used to specify the access type of the associated entity class, mapped superclass, or the embeddable class and entity attribute.

See the Access type section for more info.

# @AssociationOverride

The @AssociationOverride annotation is used to override an association mapping (e.g. @ManyToOne, @OneToOne, @OneToMany, @ManyToMany) inherited from a mapped superclass or an embeddable.

See the Overriding Embeddable types section for more info.

# @AssociationOverrides

The @AssociationOverrides is used to group several @AssociationOverride annotations.

# @AttributeOverride

The @AttributeOverride annotation is used to override an attribute mapping inherited from a mapped superclass or an embeddable.

See the Overriding Embeddable types section for more info.

# @AttributeOverrides

The @AttributeOverrides is used to group several @AttributeOverride annotations.

# @Basic

The @Basic annotation is used to map a basic attribute type to a database column.

See the Basic types chapter for more info.

# @Cacheable

The @Cacheable annotation is used to specify whether an entity should be stored in the second-level cache.

See the Caching chapter for more info.

# @CollectionTable

The @CollectionTable annotation is used to specify the database table that stores the values of a basic or an embeddable type collection.

See the Collections of embeddable types section for more info.

# @Column

The @Column annotation is used to specify the mapping between a basic entity attribute and the database table column.

See the @Column annotation section for more info.

# @ColumnResult

The @ColumnResult annotation is used in conjunction with the @SqlResultSetMapping or @ConstructorResult annotations to map a SQL column for a given SELECT query.

See the Entity associations with the named native queries section for more info.

# @ConstructorResult

The @ConstructorResult annotation is used in conjunction with the @SqlResultSetMapping annotations to map columns of a given SELECT query to a certain object constructor.

See the Multiple scalar values NamedNativeQuery with ConstructorResult section for more info.

# @Convert

The @Convert annotation is used to specify the AttributeConverter implementation used to convert the currently annotated basic attribute.

See the AttributeConverter section for more info.

# 24.1.13. @Converter

The @Converter annotation is used to specify that the current annotate AttributeConverter implementation can be used as a JPA basic attribute converter.

See the AttributeConverter section for more info.

# @Converts

The @Converts annotation is used to group multiple @Convert annotations.

See the AttributeConverter section for more info.

# @DiscriminatorColumn

The @DiscriminatorColumn annotation is used to specify the discriminator column name and the discriminator type for the SINGLE_TABLE and JOINED Inheritance strategies.

See the Discriminator section for more info.

# @DiscriminatorValue

The @DiscriminatorValue annotation is used to specify what value of the discriminator column is used for mapping the currently annotated entity.

See the Discriminator section for more info.

# @ElementCollection

The @ElementCollection annotation is used to specify a collection of basic or embeddable types.

See the Collections section for more info.

# @Embeddable

The @Embeddable annotation is used to specify embeddable types. Like basic types, embeddable types do not have any identity, being managed by their owning entity.

See the Embeddables section for more info.

# @Embedded

The @Embedded annotation is used to specify that a given entity attribute represents an embeddable type.

See the Embeddables section for more info.

# @EmbeddedId

The @EmbeddedId annotation is used to specify the entity identifier is an embeddable type.

See the Composite identifiers with @EmbeddedId section for more info.

# @Entity

The @Entity annotation is used to specify that the currently annotate class represents an entity type. Unlike basic and embeddable types, entity types have an identity and their state is managed by the underlying Persistence Context.

See the Entity section for more info.

# @EntityListeners

The @EntityListeners annotation is used to specify an array of callback listener classes that are used by the currently annotated entity.

See the JPA callbacks section for more info.

# @EntityResult

The @EntityResult annotation is used with the @SqlResultSetMapping annotation to map the selected columns to an entity.

See the Entity associations with the named native queries section for more info.

# @Enumerated

The @Enumerated annotation is used to specify that an entity attribute represents an enumerated type.

See the @Enumerated basic type section for more info.

# @ExcludeDefaultListeners

The @ExcludeDefaultListeners annotation is used to specify that the currently annotated entity skips the invocation of any default listener.

See the Exclude default entity listeners section for more info.

# @ExcludeSuperclassListeners

The @ExcludeSuperclassListeners annotation is used to specify that the currently annotated entity skips the invocation of listeners declared by its superclass.

See the Exclude default entity listeners section for more info.

# @FieldResult

The @FieldResult annotation is used with the @EntityResult annotation to map the selected columns to the fields of some specific entity.

See the Entity associations with the named native queries section for more info.

# @ForeignKey

The @ForeignKey annotation is used to specify the associated foreign key of a @JoinColumn mapping. The `@ForeignKeyannotation` is only used if the automated schema generation tool is enabled. In which case, it allows you to customize the underlying foreign key definition.

See the @ManyToOne with @ForeignKey section for more info.

# @GeneratedValue

The @GeneratedValue annotation specifies that the entity identifier value is automatically generated using an identity column, a database sequence, or a table generator. Hibernate supports the `@GeneratedValue` mapping even for UUIDidentifiers.

See the Automatically-generated identifiers section for more info.

# @Id

The @Id annotation specifies the entity identifier. An entity must always have an identifier attribute, which is used when loading the entity in a given Persistence Context.

See the Identifiers section for more info.

# @IdClass

The @IdClass annotation is used if the current entity defined a composite identifier. A separate class encapsulates all the identifier attributes, which are mirrored by the current entity mapping.

See the Composite identifiers with @IdClass section for more info.

# @Index

The @Index annotation is used by the automated schema generation tool to create a database index.

See the Columns index chapter for more info.

# 24.1.33. @Inheritance

The @Inheritance annotation is used to specify the inheritance strategy of a given entity class hierarchy.

See the Inheritance section for more info.

# @JoinColumn

The @JoinColumn annotation is used to specify the FOREIGN KEY column used when joining an entity association or an embeddable collection.

See the @ManyToOne with @JoinColumn section for more info.

# @JoinColumns

The @JoinColumns annotation is used to group multiple @JoinColumn annotations, which are used when mapping entity association or an embeddable collection using a composite identifier

# @JoinTable

The @JoinTable annotation is used to specify the link table between two other database tables.

See the @JoinTable mapping section for more info.

# @Lob

The @Lob annotation is used to specify that the currently annotated entity attribute represents a large object type.

See the BLOB mapping section for more info.

# @ManyToMany

The @ManyToMany annotation is used to specify a many-to-many database relationship.

See the @ManyToMany mapping section for more info.

# @ManyToOne

The @ManyToOne annotation is used to specify a many-to-one database relationship.

See the @ManyToOne mapping section for more info.

# @MapKey

The @MapKey annotation is used to specify the key of a java.util.Map association for which the key type is either the primary key or an attribute of the entity that represents the value of the map.

See the @MapKey mapping section for more info.

# @MapKeyClass

The @MapKeyClass annotation is used to specify the type of the map key of a java.util.Map associations.

See the @MapKeyClass mapping section for more info.

# @MapKeyColumn

The @MapKeyColumn annotation is used to specify the database column, which stores the key of a java.util.Mapassociation for which the map key is a basic type.

See the @MapKeyType mapping section for an example of `@MapKeyColumn` annotation usage.

# @MapKeyEnumerated

The @MapKeyEnumerated annotation is used to specify that the key of java.util.Map association is a Java Enum.

See the @MapKeyEnumerated mapping section for more info.

# @MapKeyJoinColumn

The @MapKeyJoinColumn annotation is used to specify that the key of java.util.Map association is an entity association. The map key column is a FOREIGN KEY in a link table that also joins the Map owner's table with the table where the Map value resides.

See the @MapKeyJoinColumn mapping section for more info.

# @MapKeyJoinColumns

The [@MapKeyJoinColumns](#) annotation is used to group
several [@MapKeyJoinColumn](#) mappings when the java.util.Map association key uses a
composite identifier.

# @MapKeyTemporal

The [@MapKeyTemporal](#) annotation is used to specify that the key of java.util.Map association is
a [@TemporalType](#) (e.g. DATE, TIME, TIMESTAMP).

See the [@MapKeyTemporal mapping](#) section for more info.

# @MappedSuperclass

The [@MappedSuperclass](#) annotation is used to specify that the currently annotated type
attributes are inherited by any subclass entity.

See the [@MappedSuperclass](#) section for more info.

# @MapsId

The [@MapsId](#) annotation is used to specify that the entity identifier is mapped by the currently
annotated `@ManyToOne` or `@OneToOne` association.

See the [@MapsId mapping](#) section for more info.

# @NamedAttributeNode

The [@NamedAttributeNode](#) annotation is used to specify each individual attribute node that
needs to be fetched by an Entity Graph.

See the [Fetch graph](#) section for more info.

# @NamedEntityGraph

The [@NamedEntityGraph](#) annotation is used to specify an Entity Graph that can be used by an
entity query to override the default fetch plan.

See the [Fetch graph](#) section for more info.

# @NamedEntityGraphs

The [@NamedEntityGraphs](#) annotation is used to group
multiple [@NamedEntityGraph](#) annotations.

# 24.1.52. @NamedNativeQueries

The @NamedNativeQueries annotation is used to group multiple @NamedNativeQuery annotations.

See the Custom CRUD mapping section for more info.

## @NamedNativeQuery

The @NamedNativeQuery annotation is used to specify a native SQL query that can be retrieved later by its name.

See the Custom CRUD mapping section for more info.

## @NamedQueries

The @NamedQueries annotation is used to group multiple @NamedQuery annotations.

## @NamedQuery

The @NamedQuery annotation is used to specify a JPQL query that can be retrieved later by its name.

See the @NamedQuery section for more info.

## @NamedStoredProcedureQueries

The @NamedStoredProcedureQueries annotation is used to group multiple @NamedStoredProcedureQuery annotations.

## @NamedStoredProcedureQuery

The @NamedStoredProcedureQuery annotation is used to specify a stored procedure query that can be retrieved later by its name.

See the Using named queries to call stored procedures section for more info.

## @NamedSubgraph

The @NamedSubgraph annotation used to specify a subgraph in an Entity Graph.

See the Fetch subgraph section for more info.

# @OneToMany

The @OneToMany annotation is used to specify a one-to-many database relationship.

See the @OneToMany mapping section for more info.

# @OneToOne

The @OneToOne annotation is used to specify a one-to-one database relationship.

See the @OneToOne mapping section for more info.

# @OrderBy

The @OrderBy annotation is used to specify the entity attributes used for sorting when fetching the currently annotated collection.

See the @OrderBy mapping section for more info.

# @OrderColumn

The @OrderColumn annotation is used to specify that the current annotation collection order should be materialized in the database.

See the @OrderColumn mapping section for more info.

# @PersistenceContext

The @PersistenceContext annotation is used to specify the `EntityManager` that needs to be injected as a dependency.

See the @PersistenceContext mapping section for more info.

# @PersistenceContexts

The @PersistenceContexts annotation is used to group multiple @PersistenceContext annotations.

# @PersistenceProperty

The @PersistenceProperty annotation is used by the @PersistenceContext annotation to declare JPA provider properties that are passed to the underlying container when the `EntityManager` instance is created.

See the [@PersistenceProperty mapping](#) section for more info.

# @PersistenceUnit

The [@PersistenceUnit](#) annotation is used to specify the `EntityManagerFactory` that needs to be injected as a dependency.

See the [@PersistenceUnit mapping](#) section for more info.

# @PersistenceUnits

The [@PersistenceUnits](#) annotation is used to group multiple [@PersistenceUnit](#) annotations.

# @PostLoad

The [@PostLoad](#) annotation is used to specify a callback method that fires after an entity is loaded.

See the [JPA callbacks](#) section for more info.

# @PostPersist

The [@PostPersist](#) annotation is used to specify a callback method that fires after an entity is persisted.

See the [JPA callbacks](#) section for more info.

# @PostRemove

The [@PostRemove](#) annotation is used to specify a callback method that fires after an entity is removed.

See the [JPA callbacks](#) section for more info.

# @PostUpdate

The [@PostUpdate](#) annotation is used to specify a callback method that fires after an entity is updated.

See the [JPA callbacks](#) section for more info.

# @PrePersist

The @PrePersist annotation is used to specify a callback method that fires before an entity is persisted.

See the JPA callbacks section for more info.

# @PreRemove

The @PreRemove annotation is used to specify a callback method that fires before an entity is removed.

See the JPA callbacks section for more info.

# @PreUpdate

The @PreUpdate annotation is used to specify a callback method that fires before an entity is updated.

See the JPA callbacks section for more info.

# @PrimaryKeyJoinColumn

The @PrimaryKeyJoinColumn annotation is used to specify that the primary key column of the currently annotated entity is also a foreign key to some other entity (e.g. a base class table in a JOINED inheritance strategy, the primary table in a secondary table mapping, or the parent table in a @OneToOne relationship).

See the @PrimaryKeyJoinColumn mapping section for more info.

# @PrimaryKeyJoinColumns

The @PrimaryKeyJoinColumns annotation is used to group multiple @PrimaryKeyJoinColumn annotations.

# @QueryHint

The @QueryHint annotation is used to specify a JPA provider hint used by a @NamedQuery or a @NamedNativeQueryannotation.

See the @QueryHint section for more info.

# @SecondaryTable

The @SecondaryTable annotation is used to specify a secondary table for the currently annotated entity.

See the @SecondaryTable mapping section for more info.

# @SecondaryTables

The @SecondaryTables annotation is used to group multiple @SecondaryTable annotations.

# @SequenceGenerator

The @SequenceGenerator annotation is used to specify the database sequence used by the identifier generator of the currently annotated entity.

See the @SequenceGenerator mapping section for more info.

# @SqlResultSetMapping

The @SqlResultSetMapping annotation is used to specify the ResultSet mapping of a native SQL query or stored procedure.

See the SqlResultSetMapping mapping section for more info.

# @SqlResultSetMappings

The @SqlResultSetMappings annotation is used to group multiple @SqlResultSetMapping annotations.

# @StoredProcedureParameter

The @StoredProcedureParameter annotation is used to specify a parameter of a @NamedStoredProcedureQuery.

See the Using named queries to call stored procedures section for more info.

# @Table

The @Table annotation is used to specify the primary table of the currently annotated entity.

See the @Table mapping section for more info.

# @TableGenerator

The @TableGenerator annotation is used to specify the database table used by the identity generator of the currently annotated entity.

See the @TableGenerator mapping section for more info.

# @Temporal

The @Temporal annotation is used to specify the `TemporalType` of the currently annotated java.util.Date or java.util.Calendar entity attribute.

See the Basic temporal types chapter for more info.

# @Transient

The @Transient annotation is used to specify that a given entity attribute should not be persisted.

See the @Transient mapping section for more info.

# @UniqueConstraint

The @UniqueConstraint annotation is used to specify a unique constraint to be included by the automated schema generator for the primary or secondary table associated with the currently annotated entity.

See the Columns unique constraint chapter for more info.

# @Version

The @Version annotation is used to specify the version attribute used for optimistic locking.