|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **OneToMany**  **(Uni)**  **Set Collection** | **OneToMany**  **(Uni)**  **List Collection** | **OneToMany**  **(Uni)**  **Map Collection** | **ManyToOne**  **(uni)** | **OneToMany**  **(Bi)** | **ManyToMany**  **(Bi)** | **OneToOne with PK** | **OneToOne with FK** |
| Parent domain class | **Collection type property** to hold one or more child objects | **Collection type property** to hold one or more child objects | **Collection type property** to hold one or more child objects | - | **Collection type property** to hold one or more child objects | **Collection type property** to hold one or more child objects |  |  |
| Child domain class | - | - | - | Parent type reference property | **Parent type reference property** | **Collection type property** to hold one or more parent objects |  |  |
| Annotation | @OneToMany  @JoinColumn | @OneToMany  @OrderColumn(to specify list index in xml )  @ListIndexBase(to specify index number/value)  @JoinColumn | @OneToMany  @JoinColumn  @MapKeyColumn | @ManyToOne | @OneToMany at parent side  @ManyToOne  At child side |  |  |  |
| Primary key | required in both side | required in both side | required in both side | required in both side |  |  |  |  |
| Foreign key | Required to build relation between two tables | Required to build relation between two tables | Required to build relation between two tables | Required to build relation between two tables |  |  |  |  |
| Orphans | possible | Possible | Possible | **Not possible** | possible | **Not possible** | Not possible | Not possible |
| cascade |  |  |  |  |  |  |  |  |
| Inverse | - | - | - | - | True | True | - | - |
| fetch |  |  |  |  |  |  |  |  |
| Lazy  (according to xml config.) | ‘true”  “false”  “extra” | ‘true”  “false”  “extra” | ‘true”  “false”  “extra” | “proxy”  ”no-proxy” |  |  | proxy”  ”no-proxy” | proxy”  ”no-proxy” |
|  |  |  |  |  |  |  |  |  |

**All annotations**

**@joincolumn**

**All Mappings**

1. **One To many**

**Cascading and lazy loading**

**Delete operations in One To Many**

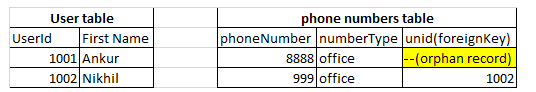
Cascading used for non select operations

**Cascading**: the non select operation performed with parent object will be reflected/propagated to its associated child object. Its not applicable for select operations.

|  |  |  |
| --- | --- | --- |
| Xml way | Annotations way | description |
| Cascade = persist |  | **Cascading persist/save operation to associated object** |
| Cascade = “save-update” |  | **Cascading save/update operation to associated object** |
| Cascade = “merge” |  | **Cascading save/update operation to associated object** |
| Cascade = “delete” |  | **Cascading save/update operation to associated object** |
| Cascade = “all” | **Cascade=CascadeType.All** | **Cascading save/update operation to associated object** |
| Cascade = “delete-orphan” |  | **Same as cascade=delete but deletes orphans records also** |
| Cascade = “all-delete-orpan” |  | **Same as cascade=all but deletes orphans records also** |
|  |  |  |

**Note:**

In One to many, if we use cascade = all ‘cascade=delete, delete operations will be reflected to associated objects also. But if we use cascade=none/persist..etc, then only paent objects will be deleted whereas child objects will remain unaffected. Only the link between child associated records with their parent records will be deeted. That link is nothing but foreign key column in child table. Such records is called as orphan records. To deete orphan records also, use cascade=”**delete-orphan**” / “**all-delete-orphan**”

****

**In annotation approach, there is not annotation for orphan removal so we use orphanRemoval=”true” of @One –to-many and @One-to-one**

**Select operations**

Lazy loading used for select operation

**Lazy loading:** In association mapping, parent object will be loaded first and child object will be load based on demand, this is called as lazy loading. Its only for select operation.

|  |  |  |
| --- | --- | --- |
| Xml way | Annotations way | description |
| Lazy=true | Performs **lazy loading** means Loads parent/main object normally but loads child/associated objects based on demand basis. | **Cascading persist/save operation to associated object** |
| Lazy=extra | **Performs extra lazy loading** means collection property of parent object will be initialized with child object only for appropriate operations. | **Cascading save/update operation to associated object** |
| Lazy=false | Performs **eager loading** means loads child/associated objects with parent/domain object irrespective of whether child objects are used or not | **Cascading save/update operation to associated object** |

**One-To-Many/Many-To-Many**

Lazy=extra(best),true(***default)***

*Cascade=all-delete-orphan*

**Many-To-One/One-To-One**

**Lazy=**no-proxy, proxy(default)

*Cascade=all*

Q1. What is the difference between **Set, List, Bag , Map**?

|  |  |
| --- | --- |
| Set | **Drawback:**   * Here we cannot store child in insertion order or in indexed order. * We can’t predict who is child of which parent class * If we want to delete the child, we have to do eager loading as we can’t access the child through indexes.   Solution is to use **LIST** collection property |
| List | **Rule:**  To maintain indexed order /insertion order, we take extra column in child table. (***index column is mandatory***)  We need to add extra annotation **@OrderColumn/** **tag <list-index>**  **Drawback:**   * Index column is manadatory * If we don’t want restriction of index colomn configuration and don’t want to store index of child in child table * If we want to access through indexes   Solution is to use **Bag** |
| Bag |  |
|  |  |
|  |  |

Map type property?

In One to many mapping, to establish a relationship, **we keep one foreign key column in child table**. And In parent class, we keep **Set/List/Map** collection type property to hold more children.

Q2. What is the difference between lazy=”proxy” and lazy=”no-proxy”?

Q3. What is the difference between <bag> and <id-bag> tag?

Q3. What is inverse and its use?

**Inverse** is used only in **bidirectional association. Inverse=”true” should be on many side of OneToMany and ManyToMany side.**

**“mappedBy” parameter is used only in bidirectional associations**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **OneToMany**  **(Uni)**  **Set Collection** | **OneToMany**  **(Uni)**  **List Collection** | **OneToMany**  **(Uni)**  **Map Collection** | **ManyToOne**  **(uni)** | **OneToMany**  **(Bi)** | **ManyToMany**  **(Bi)** | **OneToOne with PK** | **OneToOne with FK** |
| Annotation | @OneToMany  @JoinColumn | @OneToMany  @OrderColumn  @ListIndexBase | @OneToMany  @JoinColumn  @MapKeyColumn | @ManyToOne  @JoinColumn | @OneToMany at parent side  @ManyToOne  At child side  @JoinColumn | @ManyToMany at parent side  @ManyToMany  At child side | @OneToOne  @JoinColumn | @OneToOne  @JoinColumn |
| cascade | None | None | None |  |  |  |  |  |
| Inverse | - | - | - | - | True | True | - | - |
| fetch | Lazy(default)  Eager  Extra lazy  ***(@lazyCollection***  ***(lazyCollectionOption.***  ***TRUE/FALSE/EXTRA)*** | Lazy(default)  Eager  Extra lazy  ***(@lazyCollection***  ***(lazyCollectionOption.***  ***TRUE/FALSE/EXTRA)*** | Lazy(default)  Eager  Extra lazy  ***(@lazyCollection***  ***(lazyCollectionOption.***  ***TRUE/FALSE/EXTRA)*** | Eager(default) |  |  | Lazy | lazy |
| orphanRecords | True | True | True | none | none | none | none | true |