# Embedded objects

These are objects which dont have menaing ful existence of their own

for e.g holidays taken by an employee

@Embeddable

public class VacationEntry {

@Temporal(TemporalType.DATE)

private Calendar startDate;

@Column(name="DAYS")

private int daysTaken;

}

Embedded objects are stored in the same table as parent object

# Collection Mapping

Collections in entities are mapped to separate tables using

@ElementCollection(targetclass=\*\*\*\*)

if the collection types can be inferred then no need to specify target class

Either scalar types or embedded objects permitted

e.g. Set<String> nickNames

specify table name using @CollectionTable annotation

mapping Set is straight forward

mapping List specify

OrderBy

OrderColumn

mapping using Map

MapKey

MapKeyColumn

MapKeyEnumerated

MapKeyJoinColumn - if you have entities as keys

| **Map** | **Mapping** | **Key Annotation** | **Value Annotation** |
| --- | --- | --- | --- |
| Map<Basic,Basic> | @ElementCollection | @MapKeyColumn,  @MayKeyEnumerated,  @MapKeyTemporal | @Column |
| Map<Basic,Embeddable> | @ElementCollection | @MapKeyColumn,  @MayKeyEnumerated,  @MapKeyTemporal | Mapped by embeddable,  @AttributeOverride,  @AssociationOverride |
| Map<Basic,Entity> | @OneToMany, @ManyToMany | @MapKey,  @MapKeyColumn,  @MayKeyEnumerated,  @MapKeyTemporal | Mapped by entity |
| Map<Embeddable,Basic> | @ElementCollection | Mapped by embeddable,  @AttributeOverride | @Column |
| Map<Embeddable,Embeddable> | @ElementCollection | Mapped by embeddable,  @AttributeOverride | Mapped by embeddable,  @AttributeOverride,  @AssociationOverride |
| Map<Embeddable,Entity> | @OneToMany, @ManyToMany | Mapped by embeddable | Mapped by entity |
| Map<Entity,Basic> | @ElementCollection | @MapKeyJoinColumn | @Column |
| Map<Entity,Embeddable> | @ElementCollection | @MapKeyJoinColumn | Mapped by embeddable,  @AttributeOverride,  @AssociationOverride |
| Map<Entity,Entity> | @OneToMany, @ManyToMany | @MapKeyJoinColumn | Mapped by entity |