

# Hibernate Collections



Presented by Hosein Zare  
Twitter: @zare88r





# نگهداری از Collection ها

- Java Collection ها از ساختمان داده های پر استفاده در جاوا هستند که از الگوریتم های زیادی پشتیبانی میکنند
- تمامی Collection های معروف در جاوا مثل Map , Set , List و آرایه ها همگی در Hibernate پشتیبانی میشوند
- همچنین Hibernate دو Collection دیگر به نامهای bag و idbag نیز استفاده میکند



# List

- List ها ساختمان داده های ساده و راحتی برای نگهداری item ها هستند که از خصوصیات آنها میتوان به نگهداری به ترتیب عناصر استفاده کرد
- ما میتوانیم یک عنصر را در یک لیست در هر بخشی که احتیاج داشتیم اضافه کنیم و همچنین آنها را از هر index که نیاز داشتیم واکشی کنیم.



# List

```
public class Car {  
    private int id;  
    private String name = null;  
    private String color = null;  
    ...  
}
```

```
<class name="Showroom" table="SHOWROOM_LIST">  
    <id column="SHOWROOM_ID" name="id">  
        <generator class="native"/>  
    </id>  
    ...  
    <list name="cars" cascade="all" table="CARS_LIST">  
        <key column="SHOWROOM_ID"/>  
        <index column="CAR_INDEX"/>  
        <one-to-many class="Car"/>  
    </list>  
</class>
```



# Set

- Set یک ساختار داده است که عناصر آن بدون ترتیب نگهداری میشود و اجازه ی ثبت اطلاعات تکراری ندارند
- کلاس هایی که قرار است درون یک Set قرار بگیرند باید متود های hashCode و equals را override کرده باشیم



# Set

```
public class Showroom {  
    private int id = 0;  
    private String manager = null;  
    private String location = null;
```

```
// Cars are represented as set  
    private Set<Car> cars = null;
```

```
<class name="Showroom" table="SHOWROOM_SET">  
    <id column="SHOWROOM_ID" name="id">  
        <generator class="native"/>  
    </id>  
    ...  
    <set name="cars" table="CARS_SET" cascade="all">  
        <key column="SHOWROOM_ID"/>  
        <one-to-many class="Car"/>  
    </set>  
</class>
```



# نگهداری Map ها

- جایی که نیاز دارید اطلاعات را بهش شکل "کلید / مقدار" نگهداری کنید اولین انتخابتان میتواند Map باشد
- Map شبیه به یک لغت نامه میماند



# نگهداری Map ها

```
public class Showroom {  
    private int id = 0;  
    private String manager = null;  
    private String location = null;  
    private Map<String, Car> cars = null;
```

```
<map name="cars" cascade="all" table="CARS_MAP">  
    <key column="SHOWROOM_ID"/>  
    <map-key column="CUST_NAME" type="string" />  
    <one-to-many class="Car"/>  
</map>
```





# نگهداری آرایه ها

- نگهداری آرایه ها شبیه به List ها میباشد

```
public class Showroom {  
    private int id = 0;  
    private String manager = null;  
    private String location = null;  
    // List of cars  
    private String[] cars = null;  
}
```

```
<class name="Showroom" table="SHOWROOM_ARRAY">  
    <id column="SHOWROOM_ID" name="id">  
        <generator class="native"/>  
    </id>  
    ...  
    <array name="cars" cascade="all" table="CARS_ARRAY">  
        <key column="SHOWROOM_ID"/>  
        <index column="CAR_INDEX"/>  
        <element column="CAR_NAME" type="string" not-null="true"/>  
    </array>  
</class>
```



# bag

- اگر به دنبال یک Collection هستید که در آن نیازی به مرتب سازی و index ندارید میتوانید از bag استفاده کنید
- این پیاده سازی مخصوص Hibernate میباشد



# bag

```
<class name="Showroom" table="SHOWROOM_BAGS">
  <id column="SHOWROOM_ID" name="id">
    <generator class="native"/>
  </id>
  ...
  <bag name="cars" cascade="all" table="CARS_LIST">
    <key column="SHOWROOM_ID"/>
    <one-to-many class="Car"/>
  </bag>
</class>
```



# idbag

- idbag نوع دیگری از Collection هاست که به شما امکان تعریف رابطه های

```
<class name="Showroom" table="SHOWROOM_IDBAGS">
  <id column="SHOWROOM_ID" name="id">
    <generator class="native"/>
  </id>
  ...
  <idbag name="cars" cascade="all" table="SHOWROOM_CARS_IDBAGS">
    <collection-id column="SHOWROOM_CAR_ID" type="long">
      <generator class="hilo"/>
    </collection-id>
    <key column="SHOWROOM_ID"/>
    <many-to-many class="Car" column="CAR_ID"/>
  </idbag>
</class>
```




# نگهداری Collection ها با annotations

- دو راه برای تعریف روابط خارجی توسط annotation ها وجود دارد

– Foreign Key ها

– استفاده از جداول واسط




# استفاده از FK

- @OneToMany

- برای نگاشت روابط یک به چند استفاده میشود

- @JoinColumn

- نام Fk را برای نگاشت در نظر میگیرد




# استفاده از FK

```
@Entity
@Table(name="SHOWROOM")
public class Showroom {

    @OneToMany
    @Cascade(CascadeType.ALL)
    private List<Car> cars = null;
}
```

```
@Entity
@Table(name="CAR")
public class Car {
    @ManyToOne
    @JoinColumn(name="SHOWROOM_ID")
    private Showroom showroom;
}
```



# استفاده از FK

- `@OrderColumn(name="index_id")`

- جهت نگهداری ترتیب عناصر List درون یک ستون دیگر استفاده میشود

- `@MapKeyColumn`

- برای نگاشت یک map با جداول متناظر استفاده میشود





# استفاده از جدول واسط

- برای استفاده از Join Table باید یک جدول جهت نگهداری کلید های اصلی هر دو جدول بسازیم



# استفاده از جدول واسط

@Entity

```
public class Product {
```

```
    private String serialNumber;
```

```
    private Set<Part> parts = new HashSet<Part>();
```

@Id

```
    public String getSerialNumber() { return serialNumber; }
```

```
    void setSerialNumber(String sn) { serialNumber = sn; }
```

@OneToMany

@JoinTable(

```
    name="PRODUCT_PARTS",
```

```
    joinColumns = @JoinColumn( name="PRODUCT_ID"),
```

```
    inverseJoinColumns = @JoinColumn( name="PART_ID")
```