Serialize bi-directional JPA entities to JSON with jackson

I'm using Jackson to serialize my JPA model into JSON.

I have the following classes:

```
import com.fasterxml.jackson.annotation.*;
import javax.persistence.*;
import java.util.Set;

@JsonInclude(JsonInclude.Include.NON_NULL)
@JsonIgnoreProperties(ignoreUnknown = true)
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class)
@Entity
public class Parent {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  private String name;

  @JsonManagedReference
  @OneToMany(mappedBy = "parent", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
  private Set<Child> children;

  //Getters and setters
}
```

and

```
import com.fasterxml.jackson.annotation.*;
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@JsonInclude(JsonInclude.Include.NON_NULL)
@JsonIgnoreProperties(ignoreUnknown = true)
@JsonIdentityInfo(generator = ObjectIdGenerators.IntSequenceGenerator.class)
@Entity
public class Child {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  private String name;

  @JsonBackReference
  @ManyToOne
  @JoinColumn(name = "parentId")
  private Parent parent;

  //Getters and setters
}
```

I'm using the POJO mapping to serialize from model to JSON. When I serialize a Parent object I get the following JSON:

```
{
  "id": 1,
  "name": "John Doe",
  "children": [
    {
      "id": 1,
      "name": "child1"
    },{
      "id": 2,
      "name": "child2"
    }
  ]
}
```

But when I serialize a Child I get the following JSON:

```
{
  "id": 1,
  "name": "child1"
}
```

The reference to the parent is missing. Is there a way to solve this?

java json hibernate jpa jackson

asked Mar 24 '14 at 16:37

mmjmanders
997 2 7 22

isnt it bad to include UI related logic i.e json annotations in an Entity? isnt it killing modularization? – faisalbhagat Sep 12 '14 at 11:43

3 um ... no. Thats the main reason entities exist : as datamodel- representation, be it JPA, XML, JSON or even a combination of those. Having your whole app utilizing a single set of entities is an indicator for well designed app - a single set of entitities results in a single point of failure, which in return makes the app maintainable (and exchangable) to a much higher degree. – specializt Oct 14 '14 at 8:56

4 Answers

I think you have to chose between the @JsonIdentityInfo and the @JsonBackReference/@JsonManagedReference.

I would go with : @JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property="id") on your entities, removes @JsonBackReference/@JsonManagedReference pairs.

And add @JsonIgnore on the fields you want to exclude.

answered Mar 24 '14 at 18:22

GSP59
268 3 10

The problem is that use of managed/back references requires that direction of traversal is always from parent to child (that is, using managed reference first). This is a limitation for these annotations.

As the other answer suggests, use of Object Ids is the more flexible alternative that could perhaps work.

One other option that could perhaps work would be to use JSON Views or JSON Filter to conditionally include/exclude parent reference, if you can separate cases. This could get messy.

answered Mar 25 '14 at 5:56

StaxMan
64.5k 18 145 180

You can use JsonManagedReference / JsonBackReference and at the same time use JsonIdentityInfo to complement bidirectional relationships.

In question Class:

```
// bi-directional one-to-many association to Answer (Question is owner)
@JsonManagedReference
@OneToMany(mappedBy = "question", cascade = CascadeType.ALL)
@JsonIdentityInfo(generator = ObjectIdGenerators.IntSequenceGenerator.class, property =
"@QuestionAnswers")
private Set<Answer> answers = new HashSet<>();
```

In answer Class: // bi-directional many-to-one association to Question

```
@JsonBackReference
@ManyToOne
@JoinColumn(name = "questionId", referencedColumnName="id", foreignKey = @ForeignKey(name
= "fk_answer_question"))
private Question question;
```

If you need parent reference in child object, remove Managed / Back Reference, this works fine for me.

answered Jun 5 '14 at 20:34

Jorge
31 1

What is property = "@QuestionAnswers"? As far as I've understood the names of the classes are Answer and Question – Aditzu Oct 4 '16 at 17:43

You can use @JsonBackReference/@JsonManagedReference and add this method to child

```
@JsonProperty
public Long getParentId() {
    return parent == null ? null : parent.getId();
}
```

Result will be:

```
{
  "id": 1,
  "name": "child1",
  "parentId": 1
}
```

I hope this helps someone.

answered Jul 8 '16 at 15:30

liy

21 2
