2024-07-27 07:41:34.782 ERROR 4828 --- [nio-8080-exec-2] s.e.ErrorMvcAutoConfiguration$StaticView : Cannot render error page for request [/api/v1/] and exception [Could not write JSON: Infinite recursion (StackOverflowError); nested exception is com.fasterxml.jackson.databind.JsonMappingException: Infinite recursion (StackOverflowError) (through reference chain: org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.studyeasy.SpringBlog.models.Account["posts"]->org.hibernate.collection.internal.PersistentBag[0]->org.studyeasy.SpringBlog.models.Post["account"]->org.stu

ChatGPT

The error you're encountering, Infinite recursion (StackOverflowError), indicates that there is a circular reference between the Post and Account entities. When Jackson (the JSON processor) tries to serialize these entities to JSON, it falls into an infinite loop because Post references Account and Account references Post.

To solve this issue, you can use annotations to prevent the infinite recursion. You have a few options:

1. **Use @JsonIgnore**: You can ignore one side of the relationship to break the loop.

java

Copy code

package org.studyeasy.SpringBlog.models;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

public class Post {

// other fields

@ManyToOne

@JoinColumn(name = "account\_id")

@JsonIgnore

private Account account;

// getters and setters

}

1. **Use @JsonManagedReference and @JsonBackReference**: These annotations help in handling the bidirectional relationships.

java

Copy code

package org.studyeasy.SpringBlog.models;

import com.fasterxml.jackson.annotation.JsonBackReference;

@Entity

public class Post {

// other fields

@ManyToOne

@JoinColumn(name = "account\_id")

@JsonBackReference

private Account account;

// getters and setters

}

java

Copy code

package org.studyeasy.SpringBlog.models;

import com.fasterxml.jackson.annotation.JsonManagedReference;

@Entity

public class Account {

// other fields

@OneToMany(mappedBy = "account")

@JsonManagedReference

private List<Post> posts;

// getters and setters

}

1. **Use @JsonIgnoreProperties**: This annotation can be used to ignore specific properties during serialization.

java

Copy code

package org.studyeasy.SpringBlog.models;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@JsonIgnoreProperties({"posts"})

public class Account {

// other fields

@OneToMany(mappedBy = "account")

private List<Post> posts;

// getters and setters

}

Choose the method that best fits your requirements. If you need to serialize both sides of the relationship, the @JsonManagedReference and @JsonBackReference approach is often the best solution. If you only need one side, @JsonIgnore or @JsonIgnoreProperties might be simpler.

--------------------------------------------------------------------xxxx-----------------------------------------------------------------

package org.studyeasy.SpringBlog.models; import java.time.LocalDateTime; import javax.persistence.Column; import javax.persistence.Entity; import javax.persistence.GeneratedValue; import javax.persistence.GenerationType; import javax.persistence.Id; import javax.persistence.JoinColumn; import javax.persistence.ManyToOne; import javax.validation.constraints.NotBlank; import lombok.Getter; import lombok.NoArgsConstructor; import lombok.Setter; @Entity @Getter @Setter @NoArgsConstructor public class Post { @Id @GeneratedValue(strategy = GenerationType.SEQUENCE) private Long id; @NotBlank(message = "Missing post title") private String title; @Column(columnDefinition = "TEXT") @NotBlank(message = "Missing Post body") private String body; private LocalDateTime createdAt; private LocalDateTime updatedAt; @ManyToOne @JoinColumn(name="account\_id", referencedColumnName = "id", nullable = true) private Account account; }

ChatGPT

Here's how you can modify your Post model and Account model to handle the infinite recursion issue using the @JsonManagedReference and @JsonBackReference annotations.

**Post.java**

java

Copy code

package org.studyeasy.SpringBlog.models;

import java.time.LocalDateTime;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import com.fasterxml.jackson.annotation.JsonBackReference;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Getter

@Setter

@NoArgsConstructor

public class Post {

@Id

@GeneratedValue(strategy = GenerationType.SEQUENCE)

private Long id;

@NotBlank(message = "Missing post title")

private String title;

@Column(columnDefinition = "TEXT")

@NotBlank(message = "Missing Post body")

private String body;

private LocalDateTime createdAt;

private LocalDateTime updatedAt;

@ManyToOne

@JoinColumn(name = "account\_id", referencedColumnName = "id", nullable = true)

@JsonBackReference

private Account account;

}

**Account.java**

java

Copy code

package org.studyeasy.SpringBlog.models;

import java.util.List;

import javax.persistence.\*;

import com.fasterxml.jackson.annotation.JsonManagedReference;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Getter

@Setter

@NoArgsConstructor

public class Account {

@Id

@GeneratedValue(strategy = GenerationType.SEQUENCE)

private Long id;

private String username;

private String email;

@OneToMany(mappedBy = "account")

@JsonManagedReference

private List<Post> posts;

}

**Explanation**

* **@JsonBackReference**: This annotation is placed on the child entity's reference back to the parent entity. In this case, it's in the Post class on the account field. This marks the field as the back reference and prevents infinite recursion.
* **@JsonManagedReference**: This annotation is placed on the parent entity's reference to the child entity. In this case, it's in the Account class on the posts field. This marks the field as the managed reference, which is the forward part of the reference.

With these annotations, when Jackson serializes the Account object, it will include the list of Post objects. However, when it serializes each Post object, it will ignore the back reference to the Account object, thus preventing the infinite recursion.

Top of Form

Bottom of Form