**Goal:**

Deliver a schema that is according the JsonTypeInfo annotation using a wrapper object.

**Works fine for Animals:**

After adding a CustomDefinitionProvider with the following code this work pretty fine:

```
@Override
public CustomDefinition provideCustomSchemaDefinition(ResolvedType javaType,
SchemaGenerationContext context) {

    JsonTypeName = javaType.getErasedType().getAnnotation(JsonTypeName.class);
    if (jsonTypeName != null) {
        ObjectNode typeNode = context.createStandardDefinition(javaType, this);
        ObjectNode typeWrapNode = objectMapper.createObjectNode();
        typeWrapNode.set(jsonTypeName.value(), typeNode);
        return new CustomDefinition(typeWrapNode);
    }
```

The following Java POJO:

```
@JsonTypeInfo(
        use = JsonTypeInfo.Id.NAME,
        include = JsonTypeInfo.As.WRAPPER_OBJECT)
@JsonSubTypes({
        @JsonSubTypes.Type(value = Cat.class, name = "Cat"),
        @JsonSubTypes.Type(value = Dog.class, name = "Dog")
})
abstract static class Animal {
    public int furLength;
}

@JsonTypeName("Cat")
static class Cat extends Animal {
    public String miauwSound;
}

@JsonTypeName("Dog")
static class Dog extends Animal {
    public String barkSound;
}
```

Gives this schema:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "definitions": {
    "Animal": {
      "anyOf": [
        {
          "Cat": {
            "type": "object",
```

```
            "properties": {
              "furLength": {
                "type": "integer"
              },
              "miauwSound": {
                "type": "string"
              }
            }
          }
        },
        {
          "Dog": {
            "type": "object",
            "properties": {
              "barkSound": {
                "type": "string"
              },
              "furLength": {
                "type": "integer"
              }
            }
          }
        }
      ]
    }
  },
  "type": "object",
  "properties": {
    "animal": {
      "$ref": "#/definitions/Animal"
    },
    "animals": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/Animal"
      }
    },
    "someField": {
      "type": "string"
    }
  }
}
```

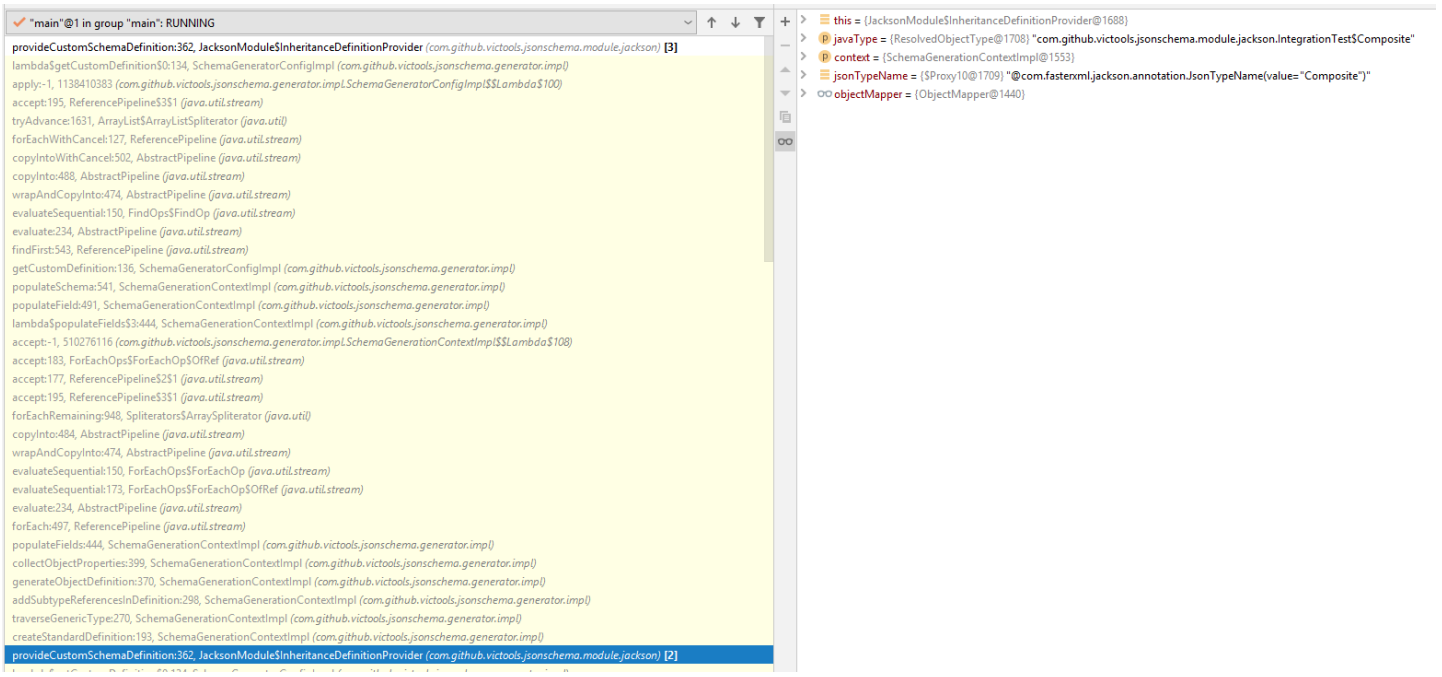
**So far so good!**

But a standard composite pattern that we are using gives problems:

The following code puts the generator in an endless recursion:

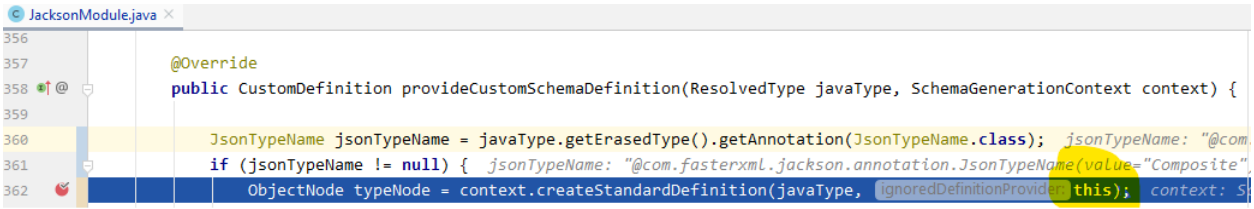
```
@JsonTypeInfo(
        use = JsonTypeInfo.Id.NAME,
        include = JsonTypeInfo.As.WRAPPER_OBJECT)
@JsonSubTypes({
        @JsonSubTypes.Type(value = Leaf.class, name = "Leaf"),
        @JsonSubTypes.Type(value = Composite.class, name = "Composite")
})
abstract static class Component {
    public String id;
    public Composite parent;
}

@JsonTypeName("Leaf")
static class Leaf extends Component {
    public String content;
}

@JsonTypeName("Composite")
static class Composite extends Component {
    public Set<Component> children;
}
```

**Frames of the stack showing one cycle of the endless recursion:**

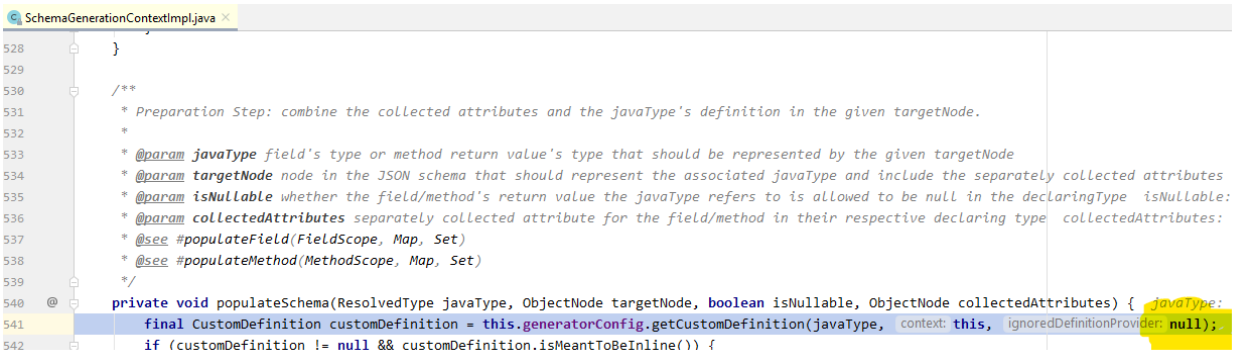


**Key elements:**

Call for the standard type definition, excluding this CustomDefinitionProvider:




Call for a definition of a type that is calling the same CustomDefinitionProvider, where this is not expected.




Please advice on a solution forward.

Regards,

Jan