Ús de relacions Lazy en aplicacions REST

A qui va dirigit

Aquest how-to va dirigit als perfils tècnics (desenvolupadors i arquitectes) que hagin de desenvolupar aplicacions amb Canigó 3.1.x amb arquitectura REST.

Versió de Canigó

Els passos descrits en aquest document apliquen a la versió del framework Canigó 3.1.x.

Introducció

Quan un objecte conté Collections entre els seus camps, per defecte aquests camps s'inicialitzen en mode Lazy. Això provoca que al accedir-hi al objecte pare a la capa web es llenci un error (*failed to lazily initialize a collection)* ja que aquestes Collections no han sigut inicialitzades.

Una manera d'evitar aquest error és usar @JsonIgnore a cada collection, però amb aquesta solució no podrem disposar de les collections en cas que les necessitem ja que sempre s'ignoraràn al crear el JSON.

Per a evitar l'error quan no els inicialitzem i alhora poder-los inicialitzar i utilitzar en cas necessari s'ha de realitzar la següent configuració.

Ús de relacions Lazy en aplicacions REST

Entitat

En aquest exemple tenim una entitat Author, que té una relació "OneToMany" amb la entitat Book

Author.java

```
…
    @OneToMany(fetch = FetchType.LAZY, mappedBy = "author")
    @JsonInclude(JsonInclude.Include.NON_DEFAULT)
    private Set<Book> books = new HashSet<Book>();
…
```

Indiquem que el mètode de càrrega de la col·lecció books és lazy.
Amb el tag JsonInclude indiquem que s'ha d'incloure aquest objecte al JSON.

Una vegada tenim configurada la entitat, hem d'indicar al dispatcher-servlet la forma de mapejar aquests objectes.

Dependències

Al pom.xml hem d'afegir la dependència a *jackson-datatype-hibernate4.*

pom.xml

```
…
<dependency>
  <groupId>com.fasterxml.jackson.datatype</groupId>
  <artifactId>jackson-datatype-hibernate4</artifactId>
  <version>2.5.3</version>
</dependency>
…
```

HibernaetAwareObjectMapper

Hem de crear la classe HibernateAwareObjectMapper per a utilitzar el mappejador en comptes del que s'utilitza per defecte al dispatcher-servlet

HibernateAwareObjectMapper

```
package cat.gencat.serveisrest.util;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.datatype.hibernate4.Hibernate4Module;

public class HibernateAwareObjectMapper extends ObjectMapper {

    public HibernateAwareObjectMapper() {
        registerModule(new Hibernate4Module());
    }
}
```

Dispatch-Servlet

Ens hem d'assegurar que es fa referència a la versió 3.2 de spring-mvc.

S'ha d'indicar al mvc:annotation-driven que s'ha d'usar la classe que hem creat (HiberanteAwareObjectMapper) per a realitzar els mapejos.

Dispatch-servlet

```
…
xsi:schemaLocation="…
http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">
…

<mvc:annotation-driven>
   <mvc:message-converters>
       <!-- Use the HibernateAware mapper instead of the default -->
       <bean
class="org.springframework.http.converter.json.MappingJackson2HttpMessageConver
ter">
        <property name="objectMapper">
                    <bean
class="cat.gencat.serveisrest.util.HibernateAwareObjectMapper" />
        </property>
        </bean>
    </mvc:message-converters>
</mvc:annotation-driven>

…

</beans>
```

Resultat

Amb aquesta configuració els objectes Lazy són inicialitzats com a null al JSON, en cas de voler tenir accés a les seves dades s'han d'inicialitzar específicament, però en cas de no necessitar-los no rebrem l'error *"failed to lazily initialize a collection"*.