Grupo 29

Item 4. Implementación de de/serialización en JSON

Manual de instrucciones

Khawla Al Asfar

Juan Carlos Utrilla Martín

Juan Rodríguez Dueñas

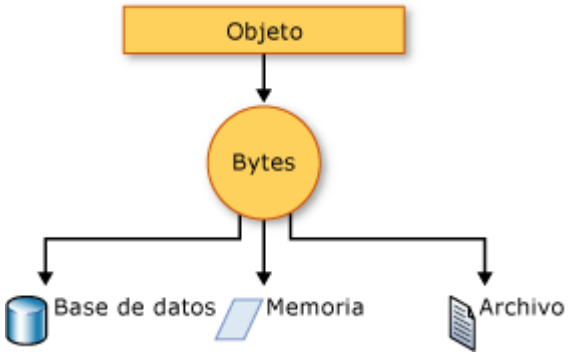
Yassine Taziny

José Manuel Lara Morilla

Contenido

1. Introducción

La serialización es el proceso de convertir una instancia de Java en una secuencia de bytes concreta. El trabajo que se describe a continuación consiste en explicar cómo aplicamos el proceso de serialización desde una clase Java a una secuencia de bytes en formato JSON y la deserialización (el proceso opuesto).



Para ello, vamos a utilizar el framework de Jackson para serializar objetos Java en JSON, y después realizaremos la deserialización, desde una cadena de texto en formato JSON, que contiene la información del objeto, a un objeto Java.

2. Preparando el framework Jackson

Para utilizar el framework de Jackson, debemos incluir las siguientes dependencias en el **pom.xml** y hacer clic derecho sobre el proyecto **Maven** > **Update project** e importamos las librerías: Core, DataBind y Annotations

```
<dependency>
	<groupId>com.fasterxml.jackson.core</groupId>
	<artifactId>jackson-core</artifactId>
	<version>2.4.2</version>
</dependency>
<dependency>
	<groupId>com.fasterxml.jackson.core</groupId>
	<artifactId>jackson-databind</artifactId>
	<version>2.4.2</version>
</dependency>
```

```
<dependency>
	<groupId>com.fasterxml.jackson.core</groupId>
	<artifactId>jackson-annotations</artifactId>
	<version>2.4.3</version>
</dependency>
```

3. UserAccount.java

Empezaremos definiendo la clase **UserAccount** que necesitamos. Ésta se encuentra en *src.main.java.security*. Aquí sólo definiremos la implementación del constructor y añadiremos anotaciones extra a la cabecera de la clase.

```
@Entity
@Access(AccessType.PROPERTY)
@JsonIgnoreProperties(ignoreUnknown = true)
public class UserAccount extends DomainEntity implements UserDetails {
```

Hemos utilizado la anotación **@JsonIgnoreProperties** que nos permite ignorar aquellas propiedades que no son relevantes en el proceso de deserialización.

```
@JsonCreator
public UserAccount(@JsonProperty("username") final String username,
                   @JsonProperty("password") final String password,
                @JsonProperty("desactivated") final boolean desactivated,
      @JsonProperty("authorities") final Collection<Authority> authorities) {
	super();
	this.username = username;
	this.password = password;
	this.desactivated = desactivated;
	this.authorities = authorities;
}
```

La anotación **@JsonCreator** la utilizamos para indicar cuál constructor queremos usar para instanciar el objeto UserAccount, comenzando con su representación JSON. Para anotar cada parámetro del constructor utilizamos la anotación **@JsonProperty** combinada con la propiedad de la clase que asigna el parámetro.

4. JacksonExample.java

La clase **JacksonExample** se utilizará para probar los procesos de serialización y deserialización. Consta de los siguientes pasos:

- **Paso 1**: configuración del ObjectMapper:

```
public static void main(final String[] args) {
      final ObjectMapper mapper = new ObjectMapper();
      mapper.configure(SerializationFeature.INDENT_OUTPUT, true);

      final SimpleDateFormat sdf = new SimpleDateFormat("dd MM yyyy");
      mapper.setDateFormat(sdf);
```

Para que no aparezca todo el JSON en una única línea, utilizamos el parámetro de configuración **SerializationFeature.INDENT_OUTPUT** para mostrar la salida de una manera más legible. En aquellos objetos que posean atributos de tipo Date, mediante un objeto tipo **SimpleDateFormat** especificaremos un formato para estas, añadiéndolo posteriormente al ObjectMapper.

- **Paso 2**: instanciación de los objetos a utilizar.

```
final UserAccount userAccount = new UserAccount();
userAccount.setId(7);
userAccount.setUsername("json");
userAccount.setPassword("466deec76ecdf5fca6d38571f6324d54");
userAccount.setDesactivated(false);

final Authority authority = new Authority();
authority.setAuthority("SPONSOR");
userAccount.addAuthority(authority);

System.out.println("INICIALIZATION OF THE OBJECT: \n\n" + userAccount);
```

En esta sección procederemos a crear el objeto a serializar y deserializar posteriormente.

- **Paso 3**: proceso de serialización. Aquí generamos el JSON a partir de una clase Java.

```
String s = null;
try {
      s = mapper.writeValueAsString(userAccount);
} catch (final JsonProcessingException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
}

System.out.println("SERIALIZATION OF THE OBJECT: \n\n" + s);
```

Con el método **writterValueAsString** convertimos el JSON a un String legible a través de la consola.

- **Paso 4**: proceso de deserialización.

```
UserAccount userAccount2 = null;
try {
      userAccount2 = mapper.readValue(s, UserAccount.class);
} catch (final JsonParseException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
} catch (final JsonMappingException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
} catch (final IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
}

System.out.println("DESERIALIZATION OF THE JSON: \n\n" +
userAccount2);
```

En este último paso, se reutilizará el String generado por la serialización para realizar el proceso de deserialización a través del método **readValue**.

5. Referencias

- **Fundamentos de la serialización**: https://msdn.microsoft.com/es-es/library/ms233836(v=vs.90).aspx
- **JSON - Wikipedia**: http://www.davismol.net/2014/12/17/json-serialization-and-deserialization-of-java-objects-with-jackson-a-concrete-example/
- **JSON serialization and deserialization of JAVA objects with Jackson**: http://www.davismol.net/2014/12/17/json-serialization-and-deserialization-of-java-objects-with-jackson-a-concrete-example/