Tarea No. 7. Data Transfer Object

Barrera Pérez Carlos Tonatihu
Profesor: José Asunción Enríquez Zárate
Web Application Development
Grupo: 3CM9

29 de enero de 2019

Índice

1. Introducción

El patrón DTO tiene como finalidad de crear un objeto plano (POJO) con una serie de atributos que puedan ser enviados o recuperados del servidor en una sola invocación, de tal forma que un DTO puede contener información de múltiples fuentes o tablas y concentrarlas en una única clase simple.

2. Desarrollo

Si bien un DTO es simplemente un objeto plano, sí que tiene que cumplir algunas reglas para poder considerar que hemos creado un DTO correctamente implementado [1]:

- Solo lectura: Dado que el objetivo de un DTO es utilizarlo como un objeto de transferencia entre el cliente y el servidor, es importante evitar tener operaciones de negocio o métodos que realicen cálculos sobre los datos, es por ello que solo deberemos de tener los métodos GET y SET de los respectivos atributos del DTO.

- Serializable: Es claro que, si los objetos tendrán que viajar por la red, deberán de poder ser serializables, pero no hablamos solamente de la clase en sí, sino que también todos los atributos que contenga el DTO deberán ser fácilmente serializables. Un error clásico en Java es, por ejemplo, crear atributos de tipo Date o Calendar para transmitir la fecha u hora, ya que estos no tienen una forma estándar para serializarse por ejemplo en Webservices o REST.

2.1. Diferencias con Value object

El patrón DTO fue llamado por error Value Object en la primera versión de Core J2EE Patterns. El nombre fue corregido en la segunda edición de este libro pero el error ya estaba hecho. Por lo que es importante aclarar las diferencias entre un DTO y un Value Object [2].

- Un DTO es un contenedor el cual es usado para trasportar datos entre capas y niveles. Principalmente contiene atributos. Se puede usar atributos publicos sin getters y setters. No contienen lógica del negocio y son serializables, lo cual es solo necesario si se transfieren DTOs a través de JVMs.

- Un Value Object representa un conjunto fijo de datos y es similar a un Java enum. Un Value Object no tiene ninguna identidad y es completamente identificado por su valor y es inmutable. Un ejemplo de esto seria Color.RED, Color.BLUE, SEX.FEMALE entre otros.

Sin embargo se siguen utilizando como sinónimos aunque esto este estrictamente mal.

3. Conclusiones

Los DTO son un patrón muy efectivo para transmitir información entre las diferentes capas de la aplicación, principalmente entre un cliente y un servidor, pues permite crear estructuras de datos independientes de nuestro modelo de datos. Además, nos permite controlar el formato, nombre y tipos de datos con los que transmitimos los datos para ajustarnos a un determinado requerimiento. Finalmente, si por alguna razón, el modelo de datos cambio (y con ello las entidades) el cliente no se afectará, pues seguirá recibiendo el mismo DTO.

Referencias

[1] O. Blancarte, "Data Transfer Object (DTO) – Patrón de diseño." `https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/`, 2018. [Online; consultado: 29-Enero-2019].

[2] A. Bien, "Value Object vs. Data Transfer Object (VO vs. DTO)." `http://www.adam-bien.com/roller/abien/entry/value_object_vs_data_transfer`, 2009. [Online; consultado: 29-Enero-2019].