**David Alejandro Abril Medina - 202224328**

**Taller 5**

**Proyecto elegido:** Apache Struts

**URL:** [**https://github.com/apache/struts.git**](https://github.com/apache/struts.git)

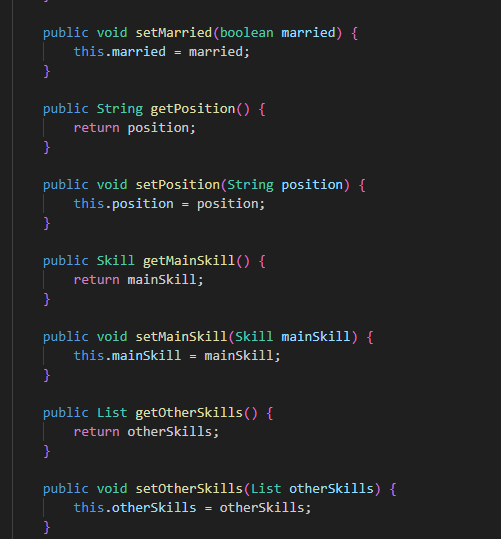
**Patron:** Chain of responsibility

Apache struts es un framework abierto basado en Java, que busca ayudar a los programadores a desarrollar aplicaciones web, el cual proporciona una infraestructura adecuada para construir aplicaciones web robustas y escalables en el futuro, la estructura general que sigue el diseño está basada en el reconocido patrón MVC, los componentes claves son; las acciones, que representan unidades lógicas de trabajo, cada acción responde a una solicitud y realiza operaciones específicas, Formularios o forms, los cuales representan los datos del usuario y actúan como un puente entre la vista y el modelo, por último configuración XML, Struts utiliza archivos de configuración XML para definir la asignación entre las URL y las acciones y tambien para lograr configurar interceptores y otros componentes.

Los retos mas grandes de diseño que enfrenta Apache Struts son, primero el manejo de estados, en aplicaciones web gestionar el estado de la aplicación y del usuario puede causar problemas, por lo que Struts hace uso de diferentes mecanismos, como los formularios y el ámbito de sesión para abordar este problema. Segundo, configurar mediante archivos XML puede volverse bastante compleja en aplicaciones de gran tamaño, por lo que gestionar la configuración es primordial para el buen uso de Apache. Por último la adaptabilidad a cambios que debe tener un framework tan grande es la adaptabilidad a los cambios que se van presentando con el transcurrir del tiempo.

Para empezar a hablar del patron, Chain of responsability, es una cadena de objetos receptores que se utilizan para construir una cadena de objetos receptores, los cuales procesan solicitudes de manera secuencial. Cada objeto tiene la capacidad de manejar la solicitud y pasarla al siguiente objeto en la cadena. Como se puede observar en el diagrama UML, en la carpeta showcase se tienen dos paquetes con diferentes clases, en Employee se inicializan todos los atributos y tambien se pueden cambiar de un empleado mediante los métodos setters, pero dentro de la clase EmployeeDao, también se puede cambiar las skills de manera más concreta haciendo uso de uno de los métodos de Employee por lo que la responsabilidad de cambiar las skills se transfiere, y se sigue transfiriendo progresivamente hasta llegar a la interfaz con el usuario, lo que demuestra el uso del patrón Chain of Responsability.

Clase Employee



Clase EmployeeDao



Diagrama UML de la relación entre los paquetes model y dao.

En este punto del proyecto tiene sentido hacer uso del patrón para, en primer lugar encapsular las diferentes clases, también para delegar responsabilidades y desacoplar el remitente del receptor, además de esto para facilitar en un futuro la adición de nuevas funcionalidades sin tener que cambiar demasiadas cosas, simplificar el código, manejar las solicitudes de manera dinámica, evitar duplicar lógica. Ahora vienen las desventajas, dentro de esas se puede observar que no existe ninguna garantía de que la solicitud será manejada en algún punto por la lógica, además de esto, entre mas grande es la cadena de responsabilidades se puede volver difícil de manejar y entender, también depende bastante de la configuración y de la limpieza de los datos, debido a que en cualquier momento el código puede colapsar si el funcionamiento por ingreso de datos de usuario no es el que se espera, por último el rendimiento puede llegar a disminuir debido al paso adicional que se debe ejecutar a través de múltiples objetos. Existen diferentes maneras para poder lograr lo mismo que el patrón chain of responsability, entre esas de las que investigue la que más me logro convencer fue el patrón command, que en vez de pasar una solicitud a través de una cadena de manipuladores, se encapsula la solicitud como un objeto command, el cual contiene información sobre como realizar una acción específica.