Diseño en capas

Índice

Capas

Se denomina “capa” de un sistema a una agrupación de componentes (clases) de iguales responsabilidades.

Si bien dijimos que una clase debe tener una responsabilidad bien definida, es muy común que tengamos múltiples clases con responsabilidades muy similares, por ejemplo, podemos tener UsuarioDAOMySQLImpl y EmpleadoDAOMySQLImpl. Ambas clases tienen una responsabilidad muy clara: implementar las operaciones de persistencia que pueden hacerse con Usuario y Empleado. Aun así, son algo diferentes (por lo pronto, alterarán el contenido de dos tablas diferentes en base de datos).

Entonces, podemos tener una “capa DAO”, “capa de persistencia” o “capa de datos” en la cual agruparemos todas las clases que administran las entidades contra la base de datos. Así podemos ir pensando en grupos de clases, formando capas en los sistemas.

Motivación

Dividir los sistemas en capas es una de las formas más comunes de organizar los sistemas hoy en día. Al dividir los componentes en capas, se ayuda a dividir el problema en partes más pequeñas y así poder reducir la complejidad total. A veces es más fácil resolver el problema en partes que tratar de resolverlo como un todo. A su vez, tener capas independientes ayuda a desacoplar las partes del sistema.

Arquitectura en n-capas

La forma más sencilla de sistema en capas es el denominado “sistema en tres capas” o “arquitectura en tres capas”. Cualquier otra arquitectura parte de esta, sub-dividiendo en dos o más cada una de estas tres capas fundamentales.

Un sistema en tres capas consiste en:

1. **Capa de presentación:** en esta capa se ubican todas las clases cuya responsabilidad sea la de presentación de los datos. Aquí van todos los componentes gráficos (pantallas por ejemplo) dedicados a mostrar los datos.

2. **Capa de aplicación o de servicios:** en esta capa va toda la denominada “lógica de negocio”. Es decir, todo lo que sea tomar los datos, procesarlos y hacer algo con ellos. Por ejemplo, mandar a la capa de datos a que los guarde. Por otro lado, también puede pedirle a la capa de datos que obtenga datos según ciertos parámetros y pasárselo a la capa de presentación para mostrarlos por pantalla.
3. **Capa de datos:** en esta capa van todas las clases encargadas de tratar con el soporte de datos (según lo que vimos, los XyzDAO y los XyzDaoImpl).

Incluso con este esquema básico ya podemos ver la ventaja de dividir el problema en capas. Supongamos que hacemos un cambio en la capa de datos. La capa de servicios “aísla” a la capa de presentación de la capa de datos. Así que si un cambio en la capa de datos, de introducir algún cambio que haga que deje de compilar el código, la capa depresentación nunca se vería afectada. De hecho, mantener las capas separadas nos permitiría reutilizar las capas enteras.

El rol de las excepciones

Es importante que si queremos tener capas independientes las responsabilidades no se mezclen. Siempre queremos mantener el acoplamiento a niveles mínimos. Es por eso que un listado desde un DAO nunca puede devolver un ResultSet. Siempre debe devolver una Collection o algún derivado. De lo contrario, no podríamos reutilizar esta capa de datos en un sistema que utilice archivos como soporte de almacenamiento (porque ResultSet es algo muy ligado a las bases de datos, de hecho, está en el paquete java.sql.*).

Algo similar ocurre con las excepciones. Imaginen un DAO cuyos métodos lanzan SQLException… Es imperioso crear excepciones o una jerarquía de excepciones que acomode la correcta separación entre las capas. Por eso, tampoco sería válido lanzar una excepción que se llame “TransaccionException” o “ClaveDuplicadaException”. Si bien son válidas para su utilización dentro de los DAO, no sería correcto que lancen esas excepciones (nuevamente, si nuestra implementación del DAO fuera basada en archivos, no tenemos el concepto de claves).

Entonces es importante crear nuestras propias excepciones, pero igual de importante es elegir el nombre correcto para que la separación entre capas sea válida conceptualmente.

Ejemplo de separación básica en capas

En el siguiente ejemplo se eligió el nombre de las excepciones para representar lo mínimo indispensable de cada capa. Es totalmente razonable que la capa de DAO arroje DAOException y que la de servicios arroje ServicioException. A su vez, la capa de presentación, que no tiene más remedio que manejar la excepción (dado que es el punto de entrada al sistema, la interacción con el usuario), en el bloque catch solo muestra una alerta anunciando el problema. Las responsabilidades de cada capa quedan bien definidas. No corresponde, bajo ninguna circunstancia, que una excepción manejada en la capa de DAO muestren una alerta dirigida hacia la pantalla.

Veamos un diagrama de cómo quedaría la separación en tres capas básicas, con el manejo de excepciones de forma tal que queden desacopladas entre sí: