Implementación del Patrón Adaptador para Integrar Librería de Analítica

Workshop / 2

Por:

Camilo Sierra

David Herrera

Angie Guevara

Jhonatan Avila

2024

Caso de Estudio: Monitoreo del Mercado Accionario con Integración de Librería de Analítica.

Introducción:

En el siguiente informe se plantea la siguiente situación *"Suponga que le asignan la tarea de diseñar una aplicación para monitorear el mercado accionario. La aplicación descarga los datos de las acciones desde múltiples fuentes en formato XML y luego presenta gráficos y diagramas para el usuario. Luego de diseñar la aplicación, le piden mejorarla integrando una librería inteligente de analítica de un tercero, sin que la aplicación quede acoplada a dicha librería para que en cualquier momento esta se pueda cambiar. La librería del tercero solo trabaja con datos en formato JSON"*, luego se analiza el diseño de una aplicación para monitorear el mercado accionario y la integración de una librería de analítica de terceros.

Diseño Inicial:

La aplicación al inicio recibe datos de las acciones en **ActionXML** en formato XML (Id, nombre, precio y tipo de moneda). Luego la aplicación cumple con su función principal de recopilar y presentar datos del mercado accionario en la clase **ActionJSON**, luego se busca la forma de pasar los datos por un patrón de diseño que nos ayude a la complejidad adaptador que realiza la conversión de XML a JSON y así finalizaría mostrando los cambios realizados

Implementación del Patrón Adaptador:

Para superar las limitaciones realizamos la búsqueda del patrón de diseño que más nos beneficie y luego implementarlo en el diseño inicial, utilizando el patrón de diseño Adaptador **AdapterXMLToJSON**. Este patrón adaptador aísla la información del código para pasar de formato de XML a JSON, permitiendo reemplazar en el futuro la información sin afectar el código principal.

Implementación del Adaptador:

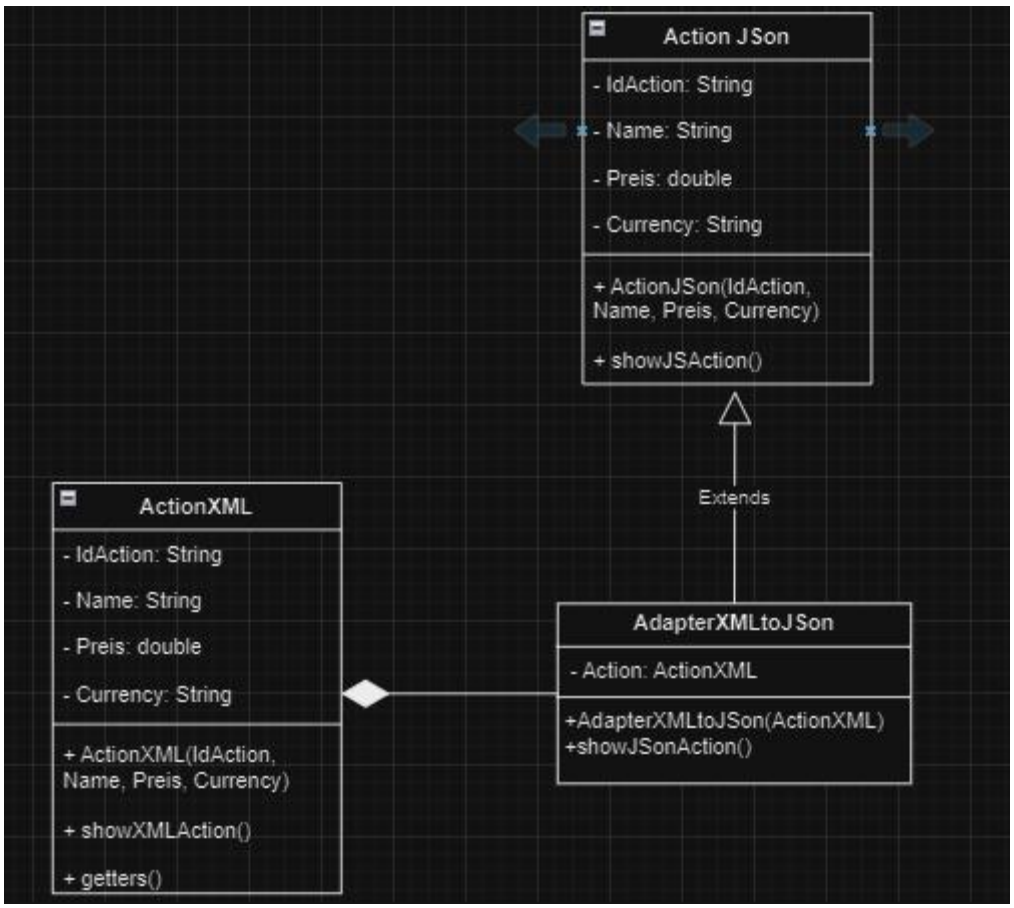
Para implementar el patrón Adaptador se:

1. Se crea la clase adaptadora **AdapterXMLToJSON** que traducen los datos XML a JSON y viceversa, y encapsulando la información.
2. la aplicación utiliza la clase **AdapterXMLToJSON** en lugar de interactuar directamente con los datos XML ce la clase **ActionXML**
3. La información queda aislada dentro de la clase adaptadora, lo que permite reemplazar sin modificar el código principal de la aplicación.
4. La clase adaptadora analiza y convierte los datos comparando la información entre las clases **ActionXMl** y **ActionJSON**, arrojando como resultado la conversión entre XML a JSON con su estructura correspondiente al formato y mostrándolo en el **Main.**

Beneficios del Patrón Adaptador:

El uso del patrón Adaptador en este caso nos permitió:

- integrar un método de conversión POO para que en el futuro sin necesidad de reescribir el código principal para que podamos adaptarlo según los datos.
- Facilita la maleabilidad del código al aislar la integración del codigo.
- Las clases adaptadoras **(AdapterJSON y AdapterXMLToJSON)** pueden ser reutilizadas para hacer el código más eficiente conciso, para que así pueda convertir la interfaz de un objeto, de forma que otro objeto pueda comprenderlo.



Main.java

```java
public class Main {
    public static void main(String[] args) {

        ActionXML apple= new ActionXML( idAction: "123", name: "Apple inc.", preis: 180, currency: "USD");
        apple.showXMLAction();

        AdapterXMLtoJSon appleJSON= new AdapterXMLtoJSon(apple);;
        appleJSON.showJSonAction();

        ActionJSon google = new ActionJSon( idAction: "123", name: "Google Inc.", preis: 210, currency: "USD");
        google.showJSonAction();
    }
}
```

ActionJSon.java

```java
public class ActionJSon {  3 usages  1 inheritor
    private String idActionJson;  2 usages
    private String name;  2 usages
    private double preis;  2 usages
    private String currency;  2 usages

    public ActionJSon(String idAction, String name, double preis, String currency) {  2 usages
        this.idActionJson = idAction;
        this.name = name;
        this.preis = preis;
        this.currency = currency;
    }


    public void showJSonAction() {  2 usages  1 override
        System.out.println("{\n" +
                "  \"idAction\": \"" + idActionJson + "\",\n" +
                "  \"name\": \"" + name + "\",\n" +
                "  \"preis\": " + preis + ",\n" +
                "  \"currency\": \"" + currency + "\"\n" +
                "}");
    }


}
```

ActionXML.java

```java
public class ActionXML {  4 usages
    private String idAction;  3 usages
    private String name;  3 usages
    private double preis;  3 usages
    private String currency;  3 usages

    public ActionXML(String idAction, String name, double preis, String currency) {  1 usage
        this.idAction = idAction;
        this.name = name;
        this.preis = preis;
        this.currency = currency;
    }

    public String getIdAction() { return idAction; }

    public String getName() { return name; }

    public double getPreis() { return preis; }

    public String getCurrency() { return currency; }

    public void showXMLAction() {  1 usage
        System.out.println("<action>\n" +
                "     <idAction><" + idAction + ">\n" +
                "     <name><" + name + ">\n" +
                "     <preis><" + preis +">\n"+
                "     <currency><" + currency + ">\n" +
                "<action>");
    }

}

```

AdapterXMLtoJSon.java × AdapterJSon.java

```java
public class AdapterXMLtoJSon extends ActionJSon{  2 usages
    ActionXML xml;  5 usages

    public AdapterXMLtoJSon(ActionXML xml) {  1 usage
        super(xml.getIdAction(), xml.getName(), xml.getPreis(), xml.getCurrency());
        this.xml = xml;
    }

@Override  2 usages
    public void showJSonAction() {
        System.out.println("The XML format was adapted to JSon format");
        System.out.println("{\n" +
                "  \"idAction\": \"" + xml.getIdAction() + "\",\n" +
                "  \"name\": \"" + xml.getName() + "\",\n" +
                "  \"preis\": " + xml.getPreis() + ",\n" +
                "  \"currency\": \"" + xml.getCurrency() + "\"\n" +
                "}");
    }
}

```

Conclusión:

El patrón Adaptador es una herramienta valiosa que podemos integrar para integrar en la aplicación de monitoreo del mercado accionario. La aplicación resultante es más flexible, escalable y mantenible, permitiendo una mejor experiencia para los usuarios y una mayor capacidad de adaptación a futuros cambios según las necesidades.