Parser

Responsables

Esteban Dib (edib@uc.cl)
Gonzalo López (gslopez@uc.cl)
Cristóbal Martínez (cnmartinez@uc.cl)
Lukas Zorich (lezorich@uc.cl)

Descripción

El parser es el componente que se preocupará de obtener la información de las distintas fuentes a partir de la búsqueda que ingrese el usuario en la web y devolverla en un formato que sea entendible para los otros componentes de la aplicación. En particular, para cada fuente buscaremos las palabras ingresadas por el usuario, y parsearemos la información obtenida a un JSON con el siguiente formato:

```
[
        {
                author: "autor", // Opcional
                date: "fecha", // Opcional
                content: "noticia, twitt, cualquier contenido",
                source:
                {
                        url: "ruta de la fuente",
                        type: "emol, twitter, gobierno_de_chile, etc",
                        extras: "cualquier otro parámetro que se quiera entregar, por
                        ejemplo, el número de followers si es un twitt" // Opcional
                }
        },
...
]
]
```

(Nos pusimos de acuerdo con el módulo de Inteligencia Artificial para el formato)

Este objeto JSON será recibido por el componente de Inteligencia Artificial para su posterior procesamiento.

Fuentes

Las fuentes de las que se obtendrá la información serán Twitter, Emol, CNN y la página de Datos del Gobierno de Chile. Esta información, se obtendrá a través de requests HTTP, ya sea a través de APIs que cada fuente exponga o parseando el HTML si no hay API disponible. La información puede venir en tres formatos: XML, JSON o HTML.

Patrones

Ya que cada fuente entrega los datos de distinta manera, utilizaremos el patrón Adapter para poder adaptar cada formato en el Json que se señaló arriba. Utilizando este patrón, es muy sencillo agregar una fuente nueva, ya que lo único que hay que hacer es agregar un nuevo adapter que herede de HTMLSourceAdapter, JsonSourceAdapter o XMLSourceAdapter según sea el formato de la fuente nueva.

Además, a partir del análisis de las distintas fuentes de noticias, nos dimos cuentas de la necesidad de utilizar el patrón Iterator para poder obtener varias páginas de noticias en las búsquedas. Y, como cada adaptador tiene una serie de pasos distintos, el patrón Template Method nos servirá para delegar implementaciones del algoritmo de parseo a las distintas subclases.

Principios Fundamentales

Abstracción

Nos abstraemos de cada adaptador concreto utilizando el SourceAdapter. Cada adaptador de una fuente en concreto se abstrae utilizando el SourceAdapter. De hecho, hay un nivel más de abstracción, ya que para cada tipo de formato (HTML, JSON o XML) hay otro adaptador que tiene el esqueleto de cómo obtener los datos de ese tipo de fuente.

Ocultamiento

Notamos que el SourceManager no sabe como se obtiene el json de los distintos componentes, tampoco sabe si las fuentes son HTML, JSON o XML, ya que todo se oculta a través del SourceAdapter y toda la implementación es ocultada por las subclases. De hecho, SourceManager solo debería poder ver el método getJson() de los distintos adaptadores.

Bajo Acoplamiento

Utilizando los distintos patrones, obtenemos bajo acomplamiento en el módulo. De hecho, si una fuente llega a cambiar, lo único que tenemos que cambiar es el adaptador concreto de la fuente, sin necesidad de cambiar más código dentro del módulo. Si necesitamos agregar más fuentes, solo agregamos un nuevo tipo de adaptador, lo que hace el módulo muy extensible.

Alta Cohesión

Cada clase es totalmente independiente del resto, ya que gracias a la herencia podemos aprovechar aquellos algoritmos que son comunes y en las clases hijas solo especificar los detalles concretos que diferencian las implementaciones entre las fuentes.

Además, el módulo es independiente de otros módulos, ya que no necesita funcionalidades de otros módulos para funcionar, y si el módulo es cambiado, no afecta a otros módulos mientras mantenga su propia interfaz.