Design Patterns: Template Method Pattern in TypeScript

Welcome to the Design Patterns in TypeScript series, which introduces some useful design patterns in web development using TypeScript.

Previous articles are as follows:

Strategy Pattern in TypeScript

Chain of Responsibility Pattern in TypeScript

Observer Pattern in TypeScript

Template Method Pattern in TypeScript

Adapter Pattern in TypeScript

Factory Method Pattern in TypeScript

Abstract Factory Pattern in TypeScript

Design patterns are very important for web developers and we can write better code by mastering them. In this article, I will use TypeScript to introduce the Template Method Pattern.

CSV (Comma-Separated Values) is a general-purpose, relatively simple file format. CSV files store tabular data (numbers and text) in plain text. When you need to process CSV data, the corresponding processing flow is shown in the following figure:

After understanding the above processing flow, let’s use Node.js to implement the function of parsing csv files.

users.csv

In the above code, we import the d3-dsv module to implement the parsing function of csv. After that, we use esno to execute the parse-cvs.ts file:

parse-csv.ts

When the above code runs successfully, the terminal will output the following result:

Markdown is a lightweight markup language that allows people to write documents in plain text format that is easy to read and write. To display Markdown documents on web pages, we must convert the Markdown documents into HTML documents.

To achieve the above functions, our processing flow is as follows:

After understanding the above processing flow, let’s use Node.js to implement the function of parsing Markdown files.

Users.md

parse-md.ts

In the above code, we import the marked module to implement the parsing function of Markdown file. After that, we use esno to execute the parse-md.ts file:

When the above code runs successfully, the terminal will output the following result:

For the previous two examples, although different types of files are parsed, you will find that their parsing process is similar.

The whole process mainly includes three steps: reading the file, parsing the file and processing the data. For this scenario, we can introduce the template method pattern to encapsulate the processing sequence of the above three steps.

The template method pattern consists of two parts: an abstract parent class and a concrete implementation subclass. Usually, the algorithm framework of the subclass is encapsulated in the abstract parent class, and it also includes the implementation of some public methods and the execution order of all methods in the encapsulated subclass. By inheriting this abstract class, subclasses also inherit the entire algorithm structure, and can choose to override the methods of the parent class.

Next, let’s look at how to implement a CSV parser and a Markdown parser using the template method pattern.

In order to better understand the following code, let’s first look at the corresponding UML class diagram:

In the above figure, we define an abstract class FileParser and then define two subclasses, CsvParser and MarkdownParser, respectively.

FileParser class

The parse method in the abstract class FileParser is the so-called template method, in which we encapsulate the process of file processing.

CsvParser class

MarkdownParser class

With the two classes CsvParser and MarkdownParser, we can parse CSV and Markdown files in the following ways:

When you successfully run the above code, the corresponding output is shown in the following figure:

Using the template method pattern, we have reimplemented the parsing of CSV and Markdown files. In fact, with the abstract class FileParser, we can easily develop different file parsers.

Finally, let’s summarize the usage scenarios of the template method pattern:

The overall steps of the algorithm are very fixed, but when individual parts are variable, the template method pattern can be used at this time to abstract the easily variable parts for subclasses to implement.

If you have any questions, please feel free to leave me a message. I will continue to introduce other patterns later, if you are interested, you can follow me on Medium or Twitter.