EasyFunc Parser

# Đặt vấn đề

## Tại sao phải cấu trúc tài liệu?

Tuy tài liệu API là một kiểu tài liệu đã được cấu trúc, tuy nhiên nó có các nhược điểm sau:

* Thông tin đưa ra còn bị trùng lắp, nhiều thông tin không mang nhiều ý nghĩa cho việc tìm kiếm và tham khảo như phiên bản, tác giả…
* Cấu trúc không cố định mà bị thay đổi tùy thuộc vào ngôn ngữ lập trình cũng như phiên bản.
* Ở nhiều định dạng khác nhau tùy thuộc vào ngôn ngữ hay phần mềm

Vì vậy, để khắc phục những nhược điểm trên chúng ta cần phải xây dựng một cấu trúc tài liệu cố định nhằm đơn giản hóa việc tương tác với tài liệu qua đó tối ưu tìm kiếm cũng như dễ dàng mở rộng hệ thống

<Hình vẽ về Adapter>

# Giải pháp EasyFunc Parser

## Ý tưởng<Not good and not sure about it, so cancel it>

Có 2 cách để hiện thực một parser.

* Cách 1:Top-down: Theo cách này chúng ta sẽ duyệt tài liệu từ trên xuống theo thành phần , với mỗi thành phần duyệt qua sẽ được gán một nhãn đã được quy định từ trước
* Cách 2:Bottom-up(????): Với cách này chúng ta sẽ dựa và các nhãn để tìm kiếm tất cả các thành phần phù hợp từ tài liệu
* Ví dụ: Chúng ta có đoạn tài liệu từ Java API docs như sau:

Chúng ta có tập nhãn gồm : name, description, annotation,keywork và others với cú pháp cho từng nhãn như sau:

public class **BufferedInputStream**

extends [FilterInputStream](file:///D:\Compressed\jdk-7u6-apidocs\docs\api\java\io\FilterInputStream.html)

A BufferedInputStream adds functionality to another input stream-namely, the ability to buffer the input and to support the mark and reset methods. When the BufferedInputStream is created, an internal buffer array is created. As bytes from the stream are read or skipped, the internal buffer is refilled as necessary from the contained input stream, many bytes at a time. The mark operation remembers a point in the input stream and the reset operation causes all the bytes read since the most recent mark operation to be reread before new bytes are taken from the contained input stream.

**Since:**

JDK1.0

Name: <keywork>? <keywork> <others> <keywork>

Với cách 1 chúng ta sẽ thu được tập dữ liệu như sau:

Name: public class **BufferedInputStream** extends [FilterInputStream](file:///D:\Compressed\jdk-7u6-apidocs\docs\api\java\io\FilterInputStream.html)

Keyword: public, class, extends

Others: **BufferedInputStream,** [FilterInputStream](file:///D:\Compressed\jdk-7u6-apidocs\docs\api\java\io\FilterInputStream.html)

Đánh giá:

* Với cách 1 chúng ta sẽ chỉ phải duyệt tài liệu một lần duy nhất tuy nhiên lại bị tốn thời gian và bộ nhớ để lưu trữ những thành phần không nằm trong tập nhãn. Với cách 2 chúng ta sẽ phải duyệt tài liệu nhiều lần nhưng kết quả lại phù hợp với mong muốn và không bị lãng phí
* Nếu tập nhãn dễ thay đổi( thêm/xóa nhãn) thì dùng cách 1 sẽ tốt hơn cách 2 do ít phải sửa code phần hiện thực hơn( chỉ cần thêm/xóa nhãn trong bước thực hiện).(Với cách 1 chúng ta có thể quét hết các khả năng, trong khi với cách 2 phải liệt kê từng bước)

## Hiện thực

# Kết luận

# Tài liệu tham khảo