XML PARSER

Edinson Sanchez
Kevin Filella
Adrian Aguilar

19 de enero de 2014

Índice

1. Introducción

Un parser, aplicado a lenguajes de programación computacionales, tiene como objetivo el de manipular ciertas cadenas de caracteres con el fin de asignar o guardar la información presentada en una estructura de datos. Una vez conocido esto, el objetivo de este proyecto es claro y conciso; el de crear un parser XML en Haskell que pueda manipular, leer y guardar los datos presentados en un archivo XML específico, con una estructura fija y con un propósito previsto; el de realizar consultas a la estructura.

1.1. Objetivo

Nuestro objetivo en este proyecto es el de crear un parser XML en Haskell que pueda manipular, leer y guardar los datos presentados en un archivo XML específico, con una estructura fija y con un propósito previsto; el de realizar consultas a la estructura.

2. Desarrollo

2.1. Desarrollo inicial

Inicialmente, debido a la naturaleza del proyecto y a las caracteristicas del curso presente, tuvimos que aprender a instalar y manejar las herraminetas de desarrollo adecuadas. Siendo este nuestro primer lenguaje completamente funcional, se nos hizo difícil, en un principio, de acostumbrarnos a no solo la sintaxis, pero también a la constante recursión que caracteriza a este lenguaje. En base a esto, más adelante presentamos los diferentes problemas que se nos presentaron a lo largo del desarrollo de nuestro proyecto.

3. Problemas

Problemas en el aprendizaje e implementación del lenguaje Haskell a la creación de un parser XML.

3.1. Primer Problema - Sintaxis

Al momento de iniciar el curso de lenguajes de programación, nosotros como estudiantes, debido al programa de estudios, hemos sido iniciados en algunos lenguajes de programación. Lenguajes como Pascal, BASIC, C, C++, Java, Android etc. Estos lenguajes son todos de alguna manera u otra muy similares entre sí. Haskell es el primer lenguaje funcional en nuestro programa de estudio y, por consiguiente, la dificultad de aprender el funcionamiento y la sintaxis del mismo fue muy elevada. Entre las dificultades del lenguajes las más prominentes fueron la sintaxis y la utilización de contadores.

3.2. Segundo Problema - Estructura

Uno de los problemas más notorios que tuvimos se dio a cabo a la hora de la asignación de los datos a la estructura. Inicialmente, aparecían errores de "match" entre tipos de datos ([String] vs [Char], IO String vs [String], entre otros). Esto lo solucionamos modificando extensivamente las funciones que manipulaban los Strings y los arreglos de Strings para que retornen los tipos de datos correctos.

3.3. Tercer Problema - Listas en Haskell

Más adelante, nos encontramos con otro problema muy tedioso, que fue el de la manipulación de listas. Aunque probamos todas nuestras funciones en consola, y funcionaban perfectamente, por alguna razón (aún desconocida para nosotros) estas funciones simplemente no funcionaban a la hora de correr nuestro programa (no se creaban las listas de Devices, Groups, Capabilities, etc). Esto lo solucionamos simplemente modificando la técnica de asignación de datos en la estructura. Creamos el tipo de dato Grupo, que contiene Device, Group y Capability; y nuestra estructura se guarda en una lista [Grupo].

3.4. Cuarto Problema - Error en creación de [Grupo]

Posterior a nuestra incómoda experiencia con el uso de listas, nos dimos cuenta que nuestra estructura (que se intentaba guardar en [Grupo]) no estaba siendo guardada correctamente. Esto fue más notorio a la hora de tratar de realizar las consultas, ya que simplemente nos lanzaba un error ("Non-exhaustive procedure", o algo parecido). La solución de este problema fue uno de los últimos cambios que hicimos en nuestro proyecto, antes de llegar a su etapa final.

4. Alcance del Proyecto

Aunque pudimos solucionar el problema propuesto, que fue el de crear un XML parser para un archivo con una estructura específica, nuestro XML parser solo funciona para ese documento, con el mismo número de niveles y nombres de atributos. Para poder utilizar este programa como un XML parser general, se le tienen que hacer extensivas modificaciones, no solo en la estructura, sino también en la lógica de asignación de los datos y la manipulación de los Strings y las listas. En cuanto a las consultas, nuestro programa realiza las búsquedas recorriendo la lista de Grupos y comparando los Strings. Además, incorpora un contador para los Devices que cumplan con la condición de búsqueda. Ingresa el nombre del ganador

5. Conclusiones

Kevin Filella: Ha decir verdad, no encuentro nada atractivo el uso del lenguaje de programación Haskell. Quizás esto se debe a que vengo de un background de lenguajes

imperativos como Java, BASIC, C, C++ y próximamente, Python. La mayor dificultad que tuve a la hora de programar en Haskell, fue entender y acostumbrarme a la recursión, seguido casi de la mano con aprender la sintaxis del lenguaje.

Edinson Sánchez: El uso del lenguaje de programación Haskell tiene bastantes ventajas y desventajas en cuanto a la capacidad de resolver problemas tal como lo hemos experimentado en este proyecto. Fue muy entretenido aprender el nuevo lenguaje. Sus características funcionales hicieron que el aprendizaje fuese muy rápido y directo. Entre las desventajas esta la poca información que arrojan los errores en tiempo de ejecución.

Adrián Aguilar: El lenguaje Haskell es bastante distinto a los demás lenguajes que hemos estudiado. Me gusto el uso extensivo de la herramienta de la recursión. La recursión le da simplicidad y elegancia al código.