Procesamiento de archivos XML en Haskell

Lozano E. , Alvarado J & .Lasso H

January 19, 2014

1 Introducción

El propósito de nuestro proyecto es que mediante un analizador de texto en este caso el parseo comprobar si un archivo xml esta bien formado y no solamente podemos comprobar si son bien formados o válidos, sino que también podemos manipularlos y trabajarlos.

El lenguaje que utilizamos fue Haskell,l es un lenguaje de programación puramente funcional, con la programación puramente funcional no decimos al computador lo que tiene que hacer, sino más bien, decimos como son las cosas.

Haskell es elegante y conciso. Se debe a que utiliza conceptos de alto nivel. Los programas Haskell son normalmente más cortos que los equivalentes imperativos. Y los programas cortos son más fáciles de mantener que los largos, además de que poseen menos errores.

2 Alcance

El alcance de nuestro proyecto es extraer la información que contiene un archivo xml, analizar su contenido , representarlo en una estructura y realizar funciones que nos permitan consultar su contenido.

3 Descripción

Se podría pensar en un documentos XML como una estructura recursiva.Lo que queremos es poder comprender en un solo vistazo las funcionalidades y/o módulos enteros.

Podemos encontrarnos con la necesidad de definir una recursividad que se puede dar en los documentos XML. En general, en cualquier documento XML, encontramos estructuras de la forma ¡f¿abc¡/f¿, en las que hay parejas de etiquetas (¡f¿ , ¡/f¿) que recursivamente pueden contener parejas (bien formadas) de etiquetas o texto plano.

Haskell incluye el soporte para los tipos de datos y funciones recursivas, listas, guardas y tuplas, el uso de este lenguaje parece ser una solución indicada para representar un documento XML.

De hecho, las combinaciones de estas características de Haskell pueden resultar en algunas funciones muy sencillas cuya versión usando lenguajes imperativos, puede llegar a resultar extremadamente tediosa de programar.

Se podría pensar en un documento XML como un conjunto de elementos, o también, como une lista de estos elementos. Por esto, la representación que se ha utilizado usando el lenguaje funcional Haskell es el de una lista.

Lo que tenemos es una Lista de Devices que tiene como elementos idD de tipo String, useragent de tipo String, fallback de tipo String y una lista de groups de tipo [Group].

La lista de Group tiene elementos de tipo Group con las siguientes características : idG de tipo String y una lista de capabilities de tipo [Capability].

La lista de Capabilities tiene elementos de tipo Caoability con estas características : name de tipo String y value de tipo String.

4 ¿Qué se implementó y qué no se implementó?

4.1 Lo implementado

Una vez extraída la información del archivo xml, la representamos en una lista de Devices que contiene elementos de tipo Device , cada Device contiene una lista de Groups que tienen elementos de tipo Group y cada Group contiene una lista de Capabilities que tienen elementos de tipo Capability. Con la lista de Devices ya obtenida podemos consultar la información de cada Device , Group o Capability.

5 Observaciones

- Aprender a programar en Haskell no fue muy sencillo, ya que para aquellos que estamos acostumbrados a lenguajes de programación imperativos es bastante abstracto,Haskell no posee estructuras de control de flujo para realizar ciertas acciones varias veces (for, while...) sino que usa recursividad.

6 Conclusiones

- Aprender el uso de nuevos lenguajes nos ayuda a simplificar la solución de problemas dificiles de resolver.

- Representar la información de cualquier archivo xml en una estructura sencilla nos ayuda hacer consultas de una manera mas eficiente.
- El objetivo se ha cumplido en su mayor parte, consistiendo en un herramienta capaz de poder evaluar consultas.