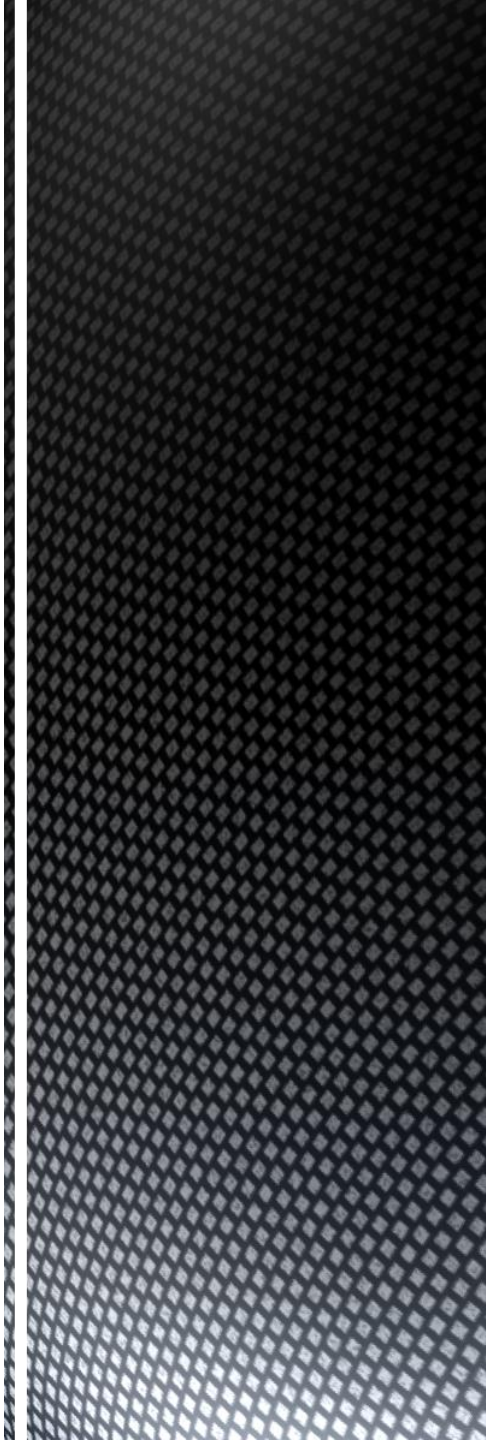
Android

Procesar XML con DOM



XML DOM

Lo básico que necesitamos para procesar con DOM es:

- Un DocumentBuilderFactory
- Un DocumentBuilder
- Un Document
- LLamar al método parse() del DocumentBuilder que se encargará de cargar el documento.
- Una función o método que dado un Document sepa recorrerlo, es decir, que conozca su arbol de tags xml.

  El método parse() cargará el Document independientemente de su estructura de nodos.

El objeto Document, una vez cargado contiene toda la información en nodos de la clase Node.

```
//Creamos el Builder y Parseamos para obtener documento
DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = fabrica.newDocumentBuilder();
Document l_document = builder.parse(l_instream);
```

XML DOM

```
<?xml version="1.0" encoding="UTF-8"?>
<citas_celebres>
  <autor nombre="Arquímedes">
            <cita>Dadme un punto de apoyo y moveré el mundo</cita>
            <url>http://es.wikipedia.org/wiki/Arqu%C3%ADmedes</url>
            <drawable>arquimedes</drawable>
  </autor>
  <autor nombre ="Confucio">
            <cita>El silencio es un amigo que jamás traiciona</cita>
            <url>http://es.wikipedia.org/wiki/Confucio</url>
            <drawable>confucio</drawable>
  </autor>

   ...

</citas_celebres>
```

Primero obtenemos la raiz del documento, en nuestro caso <citas_celebres> usando el método getDocumentElement() del objeto Document.

```
Element root = l_document.getDocumentElement()
```

Con getElementsByTagName(String tag_name) obtenemos una lista de nodos, un NodeList, que contiene todos los Nodos cuyo tag_name sea el pasado.

```
NodeList nodos_autor = root.getElementsByTagName("autor");
```

Podremos iterar por la lista de Nodos sabiendo cuantos hay con

```
nodos_autor.getLength()
```

Accediendo a cada nodo con

```
Node autor_nodo = nodos_autor.item(i);
```

XML DOM

Un Nodo puede tener atributos (Attributes) y nodos hijos ChildNodes.

Los atributos son considerados como nodos.

Podemos obtener la lista de atributos con el método `getAttributes()`

que devuelve un mapa (nombre,valor) de nodos, es decir, una colección de nodos sin orden.

```
NamedNodeMap autor_atrs = autor_nodo.getAttributes();
```

Podremos iterar por el número de atributos, que se obtiene con

```
autor_atrs.getLength()
```

para procesar cada uno de los atributos, que lo obtenemos con

```
Node theAttribute = autor_atrs.item(i);
```

Podemos saber qué atributo es preguntando por el nombre del atributo

```
String atr_name = theAttribute.getNodeName();
```

y preguntar si es el que queremos

```
atr_name.equals("nombre")
```

Una vez que sabemos que es el atributo que queremos , obtenemos su valor con

```
str_nombre = theAttribute.getNodeValue();
```

XML DOM

Un Nodo puede tener atributos (Attributes) y nodos hijos ChildNodes.

Podemos obtener la lista de nodos hijo ChildNodes de un Nodo

```
NodeList autor_childs = autor_nodo.getChildNodes();
```

y de la misma manera podremos iterar obteniendo los Node con .item() puesto que sabemos cuantos hay con .getLength()

Para determinar el nombre de un nodo usaremos .getNodeName()

Y para determinar su valor podremos usar .getNodeValue()

pero como al igual que con SAX el valor de un nodo puede ser muy largo habrá que iterar para obtener la concatenación de los valores, esto lo haremos con la siguiente función:

```
String tag_name = autor_child.getNodeName();
if (tag_name.equals("cita"))
    str_cita = obtenerTexto(autor_child);
```

```
private String obtenerTexto(Node dato)
    {
        StringBuilder texto = new StringBuilder();
        NodeList fragmentos = dato.getChildNodes();

        for (int k=0;k<fragmentos.getLength();k++)
        {
            texto.append(fragmentos.item(k).getNodeValue());
        }

        return texto.toString();
    }
```

Como Android ya no llama varias veces a un callback characters() lo que hace es construir una lista de nodos hijos para almacenar tantos fragmentos del valor del contenido como sea necesario.

XML DOM

El siguiente código es un ejemplo sobre cómo volcar el Document a un fichero xml una vez que lo hemos modificado.

```
public void escribirXML (OutputStream salida) throws Exception {
  TransformerFactory fabrica = TransformerFactory.newInstance();
  Transformer trasformador = fabrica.newTransformer();
  transformador.setOuputProperty(OutputKeys.OMIT_XML_DECLARATION,"yes");
  transformador.setOuputProperty(OutputKeys.IDENT,"yes");
  DOMSource fuente = new DOMSource(l_document);
  StreamResult resultado = new StreamResult(salida);
  transformador.transform(fuente, resultado);
}
```

Para crear un elemento podemos usar

```
Element e_autor = l_document.createElement("autor");
```

Para añadir un atributo a un elemento

```
e_autor.setAttribute("nombre","Arquímedes");
```

Para dar contenido a un elemento

```
Element e_cita = l_document.createElement("cita");
Text texto_cita = l_document.createTextNode("mitexto")
e_cita.appendChild(texto_cita);
e_autor.appendChild(e_cita);
```

Para obtener el Raiz y añadirle el nuevo elemento creado

```
Element root = l_document.getDocumentElement()
root.appendChild(e_autor);
```

Habrá que iterar tantas veces como autor y añadir tantos hijos y atributos como sea necesario.

El nodo raiz se crea como los demás elementos con createElement() y debería ser el primero que se crea tras la instanciación del nuevo objeto Document. Es decir en vez de llamar a builder.parse() comenzamos a añadir elementos en el nuevo xml document