TigaseDoc

Release 0.1

Tigase, Inc.

Dec 16, 2022

CONTENTS

CHAPTER

ONE

OVERVIEW

Tigase XML Tools is a library providing support for fast and efficient parsing and working with XML documents.

It contains of many classes however three of them, described below, are the most important.

1.1 Element

This class represent single XML element. Contains element name, namespace, attributes and inner elements if any. Instances of this class are mutable and not synchronized, so it is required to make sure that only single thread will work on particular instance of the `Element` class.

> **Warning:** Methods which name contains `StaticStr` require that passed parameters are static strings, which mean that strings needs be static or result of `String::intern()` method. This requirement is a result of usage == instead of `.eqauls()` for comparison inside this methods which make this comparison faster.

1.1.1 Creating new instance

To create new element instance one of a few constructors may be used. Each of them require as a first argument the name of element.

1.1.2 Attributes

Element attributes are easily accessing using one of following methods:

`String getAttributeStaticStr(String attName)`
: Method returns attribute value for passed attribute name. It will return `null` if attribute is not set.

`Map<String, String> getAttributes()`
: Method returns a map of attributes which are set for this XML element.

You may easily modify attribute values by using one of following methods:

`void setAttribute(String key, String value)`
: Set value for the attribute. Does not support `null` value. To remove a value for attribute, you need to use `removeAttribute()` method.

`void setAttributes(Map<String, String> newAttributes)`
: Sets attributes for element to attribute and values passed in provided map.

`void removeAttribute(String key)`
: Removes attribute and its value from element attributes.

1.1.3 Children

Each instance of the `Element` class may contain elements inside it (inner elements) named here children. To access them you may call:

`Element getChild(String name); Element getChild(String name, String child_xmlns); Element getChildStaticStr(String name); Element getChildStaticStr(String name, String child_xmlns)`
Returns a child element or `null`

`List<Element> getChildren()`
Returns a list of children elements or `null`

Note: Each of this methods may return a null if there is no child matching requirements.

To add elements as a children of the element call `void addChild(XMLNodeIfc child)` or `void addChildren(List<Element> children)`. To remove elements, you need to retrieve instance of the `Element` which you want to removed and call `boolean removeChild(Element child)`.

1.1.4 Value

In XML each element may have value assigned. To retrieve elements value you need to call `String getCData()` and to set elements value `void setCData(String argCData)`.

1.2 DomBuilderHandler

This class is an implementation of `SimpleHandler` interface, which is responsible for creation of elements and building XML trees in response to its method calls made by `SimpleParser` (XML parser).

1.3 SimpleParser

It is an implementation of a XML parser which is responsible for parsing provided array of chars and calling instance of `SimpleHandler` to react on element being read, etc.

CHAPTER

TWO

USAGE

2.1 Parsing XML

```
import tigase.xml.*;

DomBuilderHandler domHandler = new DomBuilderHandler();
SimpleParser parser = SingletonFactory.getParserInstance();

// array of chars to parse
char[] data = "<test/>".toCharArray();

// parsing data using parser and handler
parser.parse(handler, data, 0, data.length);

// check if there was no pasing errors
if (domHandler.parseError()) {
    // do something if XML parsing fails, ie. due to invalid characters in the input␣
↪array..
}

// retrieve queue of parsed elements (root elements)
Queue<Element> elems = domHandler.getParsedElements();

// for each parsed element print it
Element elem = null;
while ((elem = elems.poll()) != null) {
    System.out.println("parsed element = " + elem);
}
```

2.2 Creating elements tree

Creating `message` element with `body` inner element. Body element will contain a value `Test`.

Example.

```
import tigase.xml.*;

Element messageElem = new Element("message");
Element bodyElem = new Element("body");
```

(continues on next page)

(continued from previous page)

```
bodyElem.setCData("Test");
messageElem.addChild(bodyElem);

System.out.println(messageElem.toString());
```

Result.

```
<message><body>Test</body></message>
```

2.3 Modifying elements

In `messageElem` variable we have a message element created in a previous example. Now we will set `message` attribute `id` to `1`, remove `body` inner element and add new element `test`.

Example.

```
import tigase.xml.*;

messageElem.setAttributeStaticStr("id", "1");

Element bodyElem = messageElem.getChildStaticStr("body");
if (bodyElem != null) {
    messageElem.removeChild(bodyElem);
}

Element testElem = new Element("test");
messageElem.addChild(testElem);

System.out.println(messageElem.toString());
```

Result.

```
<message id="1"><test/></message>
```

2.4 Serializing to XML

To serialize an element and its subelements to `String` you need to call its `toString()` method which will return serialized element.