XML & JSON Parsing

XML

- XML stands for **Extensible Mark-up Language.** XML is a very popular format and commonly used for sharing data on the internet.
- We can parse the xml document by DOM parser, SAX parser, XMLPullParser etc.
- DOM can be used to create and parse the xml file but SAX parser can only be used to parse the xml file.
- DOM consumes more memory than SAX.
- XMLPullParser is very efficient, easy to use and consumes less memory.

XML elements

- An XML file consists of a number of elements that together make an XML file. They are the basic building blocks of XML document. These elements are used to store some text elements, attributes, media, etc. Below is the syntax of XML element:

<element-name attributes> Contents.. </element-name>

- Here, **element-name** is the name of the element and attributes are used to define the property of XML element.

Android XML Parsing using DOM Parser

- Here **<name>** and **<address>** are xml elements.

```
<?xml version="1.0"?>
<records>
    <Student>
    <name>Bharat Patel</name>
    <address>Surat</address>
    </Student>
    <Student>
    <name>Manish Joshi</name>
    <address>Navsari</address>
    </Student>
    <Student>
    <name>Ravi Gupta</name>
    <address>Ahmedabad</address>
    </Student>
</records>
```

```
InputStream is = getAssets().open(“Data.xml");

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(is);

Element element = doc.getDocumentElement();
element.normalize();

NodeList nList = doc.getElementsByTagName(“Student");
for (int i = 0; i < nList.getLength(); i++) {
    Node node = nList.item(i);
    if (node.getNodeType() == Node.ELEMENT_NODE) {
    Element element2 = (Element) node;
    tv1.setText(tv1.getText()+"\nName : " + getValue("name", element2)+"\n");
    tv1.setText(tv1.getText()+“Address : " + getValue(“address", element2)+"\n");
    tv1.setText(tv1.getText()+"----------------------");
    }
```

```
private static String getValue(String tag, Element element)
 {
    NodeList nodeList = element.getElementsByTagName(tag)
                                .item(0).getChildNodes();
    Node node = (Node) nodeList.item(0);
    return node.getNodeValue();
}
```

JSON Parser

JSON

- **JSON** (JavaScript Object Notation) is a programming language .
- It is open standard designed for human readable data interchange.
- It is minimal, textual, and a subset of JavaScript.
- It is an alternative to XML.
- Android provides support to parse the JSON object and array.
- Android provides four different classes to manipulate JSON data. These classes are **JSONArray, JSONObject, JSONStringer and JSONTokenizer.**

JSON Uses

- It used when writing java script based application.
- It is used for serializing and transmitting data over network connection.
- Mainly used to transmit data between server and web application.
- It can be used with modern programming languages.
- Used by web services and APIs to provide public data.

Advantage of JSON over XML

1) JSON is faster and easier than XML.
2) Unlike XML, it is shorter and quicker to read and write.
3) It can use array.
4) JSON is light weighted, structured and is an independent data exchange format that is used to parse data.
5) It is free to open use and open-source tool.
6) In JSON value retrieval is easy.

JSON Elements

Sr.No	Component & description
1	**Array([)** In a JSON file , square bracket ([) represents a JSON array.
2	**Objects({)** In a JSON file, curly bracket ({) represents a JSON object.
3	**Key** A JSON object contains a key that is just a string. Pairs of key/value make up a JSON object.
4	**Value** Each key has a value that could be string , integer or double etc.

JSON Object

- A JSON object contains key/value pairs.
- The keys are strings and the values are the JSON types.
- Keys and values are separated by comma. The { (curly brace) represents the json object.

```
{
   “Student": {
      ”name”:      “Bharat",
      ”Address”:   ”Surat”
    }
}
```

JSON Object

- For parsing a JSON object, an object of class JSONObject is created and specify a string containing JSON data to it. Its syntax is –

```
JSONObject stud = new JSONObject(JSON_STRING)
                                .getJSONObject(“Student");
```

- The method **getJSONObject** returns the JSON object. The method **getString** returns the string value of the specified key.

```
String studname=stud.getString("name");
String studaddress=stud.getString(“Address");
```

```
public static final String JSON_STRING="{\“Student\":{\"name\":\“Bharat
  \", \“Address\":Surat}}";
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
TextView tv=(TextView)findViewById(R.id.textView1);
try {
JSONObject stud= new JSONObject(JSON_STRING)
                                        .getJSONObject(“Student");
String studname = stud.getString("name");
String studaddress = stud.getString(“Address");
String str = “Student Name:"+ studname + "\n“ + “Student Address:“ +
  studaddress;
tv.setText(str);
} catch (Exception e) {e.printStackTrace();}}
```

JSON Array

- In the JSON file, you can put the data in the form of JSON array i.e. it behaves like simple array i.e. you can store data one after the other and can access that data using a zero-based indexing.
- Inside the JSON array, you can put values like String, boolean, Integer, float, etc. Apart from these, you can put some other JSON array or JSON object in a particular JSON array.

JSON Array

- The [(square bracket) represents the json array.

  ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

- An example of json array.

```
{ “Student" :
    [
      {"id":"1","name":“Bharat Patel",“Address":“Surat"},
      {"id":"2","name":“Manish Joshi",“Address":“Navsari"}
    ]
}
```

```
TextView output = (TextView) findViewById(R.id.textView1);
String strJson="{ \“Student\" :[{\"id\":\"101\",\"name\":\“BharatPatel\",\“Address\":\“Surat\
  "}, {\"id\":\"102\",\"name\":\“Manish Joshil\",\“Address\":\“Navsari\"}] }";
    String data = "";
        try {
             // Create the root JSONObject from the JSON string.
            JSONObject  jsonRootObject = new JSONObject(strJson);
            //Get the instance of JSONArray that contains JSONObjects
             JSONArray jsonArray = jsonRootObject.getJSONArray(“Student");
            //Iterate the jsonArray and print the info of JSONObjects
             for(int i = 0; i < jsonArray.length(); i++){
               JSONObject jsonObject = jsonArray.getJSONObject(i);
               int id = Integer.parseInt(jsonObject.getString("id").toString());
               String name = jsonObject.getString("name").toString();
               String  address = jsonObject.getString(“address").toString();
               data += ”id = "+ id +" \n Name= "+ name +" \n Address= "+ address +" \n ";
             }
             output.setText(data);
          } catch (JSONException e) {e.printStackTrace();} }
```

Volley

Volley

- Volley is a library that makes networking for Android apps easier and most importantly, faster.
- Volley has two classes that you will have to deal with:
- **RequestQueue** : Requests are queued up here to be executed.
- **Request (and any extension of it)** : Constructing a network request. This class contains the necessary information to make API requests.

Volley

- A Request object comes in three major types:
- **JsonObjectRequest** : To send and receive JSON Object from the server
- **JsonArrayRequest** : To receive JSON Array from the server
- **ImageRequest** : To receive an image from the server
- **StringRequest** : To retrieve response body as String (ideally if you intend to parse the response by yourself)

StringRequest

```
String url = "http://url/";
StringRequest stringRequest = new StringRequest(
        Request.Method.GET, url,
         new Response.Listener() {
            @Override
            public void onResponse(String response)
             {    }
         },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error)
             {     }
        });
requestQueue.add(stringRequest);
```

JSONObjectRequest

```
String url = "http://url/";
JsonObjectRequest  jsonObjectRequest = new JsonObjectRequest(
        Request.Method.GET, url, null,
        new Response.Listener() {
            @Override
            public void onResponse(JSONObject response)
            {    }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error)
            {    }
        });
requestQueue.add(jsonObjectRequest);
```

JSONArrayRequest

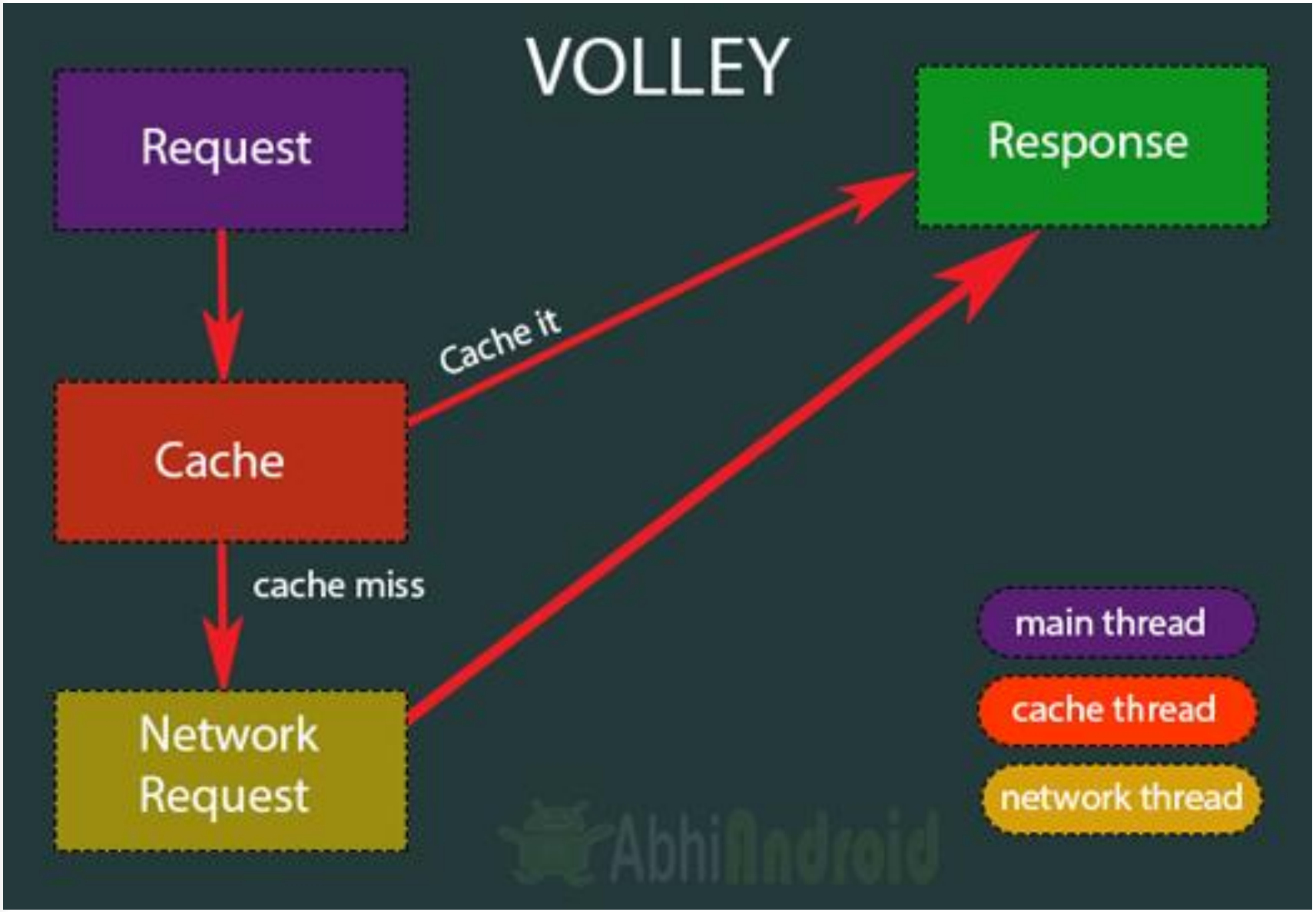
```
String url = "http://url/";
JsonArrayRequest  jsonArrayRequest = new JsonArrayRequest (
        Request.Method.GET, url, null,
        new Response.Listener() {
            @Override
            public void onResponse(JSONArray response)
            {    }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error)
            {    }
        });
requestQueue.add(jsonArrayRequest );
```

Volley Uses/Advantages

- It can manage the caching and processing of network requests.
- It helps cache and memory management.
- It manages request queuing and prioritization.
- It enables customization of the library to fit our needs.
- It saves time from writing the same network/cache code over and over.

Volley Uses/Advantages

- It can run many concurrent network connections.
- It provides tracing and debugging tools.
- It enables the automatic scheduling of network connections.
- It enables caching.
- It supports OkHttp.


VOLLEY
Request
Response
Cache it
Cache
cache miss
Network Request
main thread
cache thread
network thread
AbhiAndroid