JSON WITH JAVA

This tutorial will teach you how to encode and decode JSON objects using Java programming language. Let's start with preparing environment to start our programming with Java for JSON.

Environment

Before you start with encoding and decoding JSON using Java, you will need to install any of the JSON modules available. For this tutorial I downloaded and installed JSON.simple and add the location of **json-simple-1.1.1.jar** file to environment variable CLASSPATH:

Mapping between JSON and Java entities

JSON.simple maps entities from the left side to the right side while decoding or parsing, and maps entities from the right to the left while encoding.

JSON	Java
string	java.lang.String
number	java.lang.Number
true|false	ava.lang.Boolean
null	null
array	java.util.List
object	java.util.Map

While decoding, default concrete class of *java.util.List* is *org.json.simple.JSONArray* and default concrete class of *java.util.Map* is *org.json.simple.JSONObject*.

Encoding JSON in Java

Following is a simple example to encode a JSON object using Java JSONObject which is a subclass of java.util.HashMap. No ordering is provided. If you need strict ordering of elements use JSONValue.toJSONString(map) method with ordered map implementation such as java.util.LinkedHashMap.

```
import org.json.simple.JSONObject;

class JsonEncodeDemo
{
   public static void main(String[] args)
   {
      JSONObject obj = new JSONObject();

      obj.put("name", "foo");
      obj.put("num", new Integer(100));
      obj.put("balance", new Double(1000.21));
      obj.put("is_vip", new Boolean(true));

      System.out.print(obj);
   }
}
```

While compile and executing above program, this will produce following result:

```
{"balance": 1000.21, "num":100, "is_vip":true, "name":"foo"}
```

Following is another example which shows JSON object streaming using Java JSONObject:

```
import org.json.simple.JSONObject;

class JsonEncodeDemo
{
   public static void main(String[] args)
   {
      JSONObject obj = new JSONObject();

      obj.put("name","foo");
      obj.put("num",new Integer(100));
      obj.put("balance",new Double(1000.21));
      obj.put("is_vip",new Boolean(true));

      StringWriter out = new StringWriter();
      obj.writeJSONString(out);

      String jsonText = out.toString();
      System.out.print(jsonText);
   }
}
```

While compile and executing above program, this will produce following result:

```
{"balance": 1000.21, "num":100, "is_vip":true, "name":"foo"}
```

Decoding JSON in Java

Following example makes use of **JSONObject** and **JSONArray** where JSONObject is a java.util.Map and JSONArray is a java.util.List, so you can access them with standard operations of Map or List.

```
import org.json.simple.JSONObject;
import org.json.simple.JSONArray;
import org.json.simple.parser.ParseException;
import org.json.simple.parser.JSONParser;

class JsonDecodeDemo
{
   public static void main(String[] args)
   {
      JSONParser parser=new JSONParser();
      String s = "[0,{\"1\":{\"2\":{\"3\":{\"4\":[5,{\"6\":7}]}}}}]";
      try{
         Object obj = parser.parse(s);
         JSONArray array = (JSONArray)obj;
         System.out.println("The 2nd element of array");
         System.out.println(array.get(1));
         System.out.println();

         JSONObject obj2 = (JSONObject)array.get(1);
         System.out.println("Field \"1\"");
         System.out.println(obj2.get("1"));

         s = "{}";
         obj = parser.parse(s);
         System.out.println(obj);

         s= "[5,]";
         obj = parser.parse(s);
         System.out.println(obj);

         s= "[5,,2]";
         obj = parser.parse(s);
         System.out.println(obj);
      }catch(ParseException pe){
         System.out.println("position: " + pe.getPosition());
         System.out.println(pe);
      }
```

```
    }
}
```

While compile and executing above program, this will produce following result:

```
The 2nd element of array
{"1":{"2":{"3":{"4":[5,{"6":7}]}}}}

Field "1"
{"2":{"3":{"4":[5,{"6":7}]}}}
{}
[5]
[5,2]
```