Migrate from json-lib to jackson

json-lib is a dead project, is very slow, is a memory hog, and has memory leaks. We will migrate to jackson.

In 4.10.1+ and 5.7.0+ Jackson marshaling is used by default but will failover to json-lib if invalid attributes in JSON are used.

How to migrate to jackson (using maps)

Example: https://github.com/Internet2/grouper/commit/ef4961ebf298b06f366217f73468138089dfe8e9

Old way with json-lib

FROM

```
JSONObject jsonObject = (JSONObject) JSONSerializer.toJSON( body );
```

That has an object model that is obviously different than jackson, but it is similar. Convert to map instead, or use the jackson parsing object model. In this case we will try the object model

TO

```
JsonNode jsonObject = GrouperUtil.jsonJacksonNode(body);
```

Change calls, e.g.

FROM

```
      debugMap.put("totalSize", jsonObject.getString("totalSize"));
```

TO

```
      debugMap.put("totalSize", GrouperUtil.jsonJacksonGetString(jsonObject, "totalSize"));
```

FROM

```
        String userId = jsonObjectUser.getString("userId");
```

TO (be careful about NPE if getting ints!)

```
        String userId = GrouperUtil.jsonJacksonGetString(jsonObjectUser, "userId");
```

FROM

```
        JSONObject userWithGroupsJson = new JSONObject();
        JSONArray groupsJson = new JSONArray();
```

TO

```
        ObjectNode userWithGroupsJson = GrouperUtil.jsonJacksonNode();
        ArrayNode groupsJson = GrouperUtil.jsonJacksonArrayNode();
```

FROM

```
jsonObject.containsKey("id")
```

TO

```
jsonObject.has("id")
```

FROM

```
jsonObject.getArrayNode("groups")
```

TO

```
(ArrayNode)jsonObject.get("groups")
```

FROM

```
      jsonObject.put("name", name);
```

TO

```
      GrouperUtil.jsonJacksonAssignString(jsonObject, "name", name);
```

FROM

```
jsonObject.toString();
```

TO

```
GrouperUtil.jsonJacksonToString(jsonObject);
```

Benchmark jackson vs json-lib

```
package edu.internet2.middleware.grouper.app.remedy.digitalMarketplace;

import java.io.File;
import java.util.List;
import java.util.Map;

import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

import edu.internet2.middleware.grouper.util.GrouperUtil;
import net.sf.json.JSONArray;
import net.sf.json.JSONObject;
import net.sf.json.JSONSerializer;

public class Test {

  public Test() {
  }

  public static void main(String[] args) throws Exception {

    String json = GrouperUtil.readFileIntoString(new File("c:/temp/file.json"));

    JSONObject jsonObject = (JSONObject) JSONSerializer.toJSON( json );
    JSONArray jsonArray = jsonObject.getJSONArray("data");

    long startNanos = System.nanoTime();

    jsonObject = (JSONObject) JSONSerializer.toJSON( json );
    jsonArray = jsonObject.getJSONArray("data");

    System.out.println("json-lib took: " + ((System.nanoTime()-startNanos)/1000000) + "ms, found " + jsonArray.
size() + " records" );

    //create ObjectMapper instance
    ObjectMapper objectMapper = new ObjectMapper();
    //read JSON like DOM Parser
    JsonNode rootNode = objectMapper.readTree(json);
    JsonNode dataNode = rootNode.path("data");

    startNanos = System.nanoTime();

    rootNode = objectMapper.readTree(json);
    dataNode = rootNode.path("data");

    System.out.println("jackson took: " + ((System.nanoTime()-startNanos)/1000000) + "ms, found " + dataNode.
size() + " records" );

    startNanos = System.nanoTime();

    Map<String, Object> map = objectMapper.readValue(json, new TypeReference<Map<String,Object>>(){});

    List data = (List)map.get("data");

    System.out.println("jackson map took: " + ((System.nanoTime()-startNanos)/1000000) + "ms, found " + data.
size() + " records" );

  }

}
```

Results

```
json-lib took: 3081ms, found 2000 records
jackson took: 14ms, found 2000 records
jackson map took: 51ms, found 2000 records
```

The problem

First problem is Penn's daemon keeps crashing when running the remedy digital marketplace job that uses json-lib. Not sure what changed, maybe there are more users to download now?

For the remedy digital marketplace connector, it downloads all users in batches of 2000. This is a 2Meg json string. It takes 1.5 seconds to do the WS call from the app. Then it takes 2 minutes to convert that into a JSON object.

```
2020-11-12 11:51:55,332: [main] DEBUG GrouperDigitalMarketplaceLog.marketplaceLog(43) - - method:
retrieveDigitalMarketplaceUsersHelper, authnMillis: 229ms, getMillis: 1511ms, totalSize: 128379, first:
43253.0, last: zzimm, elapsedMillis: 129512
```

Heres a memory leak report
https://sourceforge.net/p/json-lib/bugs/search/?q=memory
New provisioning connectors shouldnt use it. Eventually we need to move away from it for web services too.

If I set the paging size down to 200 instead of 2000 it seems like it might work, but we still need to swap this out. 340ms to make rest call and 70ms to parse json...

```
2020-11-12 12:24:34,113: [main] DEBUG GrouperDigitalMarketplaceLog.marketplaceLog(43) - - method:
retrieveDigitalMarketplaceUsersHelper, getMillis: 340ms, totalSize: 128379, first: aahuja77, last: zarany,
elapsedMillis: 409
```

well, its not consistent. sometimes it takes longer. this says 300ms to call WS and 2.5 seconds to parse json 😒

```
2020-11-12 12:27:24,069: [main] DEBUG GrouperDigitalMarketplaceLog.marketplaceLog(43) - - method:
retrieveDigitalMarketplaceUsersHelper, getMillis: 298ms, totalSize: 128379, first: 43256.0, last: zyuqing,
elapsedMillis: 2950
```

Its going up by a factor of 10 as it runs
Here is 300ms for call, and 13 seconds to parse 🙂

```
2020-11-12 12:37:05,796: [main] DEBUG GrouperDigitalMarketplaceLog.marketplaceLog(43) - - method:
retrieveDigitalMarketplaceUsersHelper, getMillis: 369ms, totalSize: 128379, first: adambis, last: zampou,
elapsedMillis: 13262
```

Heres a datapoint parsing the 2meg json

```
json-lib took: 3471ms, found 2000 records
jackson took: 180ms, found 2000 records
```

Other options

com.googlecode.json-simple
https://search.maven.org/artifact/org.json/json/20140107/jar (or newer)
And I know I use jackson around here somewhere too….
FWIW: I just added a new CLC that is using json-lib to my custom stack…
So… Please make it clear when that dependency evaporates. (I will likely need to convert away from json-lib in a project or two around here….)
Note I have not done any performance/feature comparisons against the libs… YMMV. (edited)

OK, thanks for the pointer. That one looks equivalent to jackson, and we already have jackson, so Im inclined to keep using that one. I dont know when json-lib would be taken out of grouper, you have some time https://www.overops.com/blog/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/