Javabin

javabin is a custom binary format used to write out solr's response in a fast and efficient manner. SOLR-486 is the jira issue .

Currently only a java library is available to write/read this format . This is the default format used by SolrJ and this is the format used for inter-Solr communication in distributed search. Use `wt=javabin` in request parameter to get the output in this format.

Supported Types

It supports all the data types needed in Solr Response

- null
- java.lang.String
- boolean
- byte
- short
- double
- int
- long
- float
- java.util.Map
- org.apache.solr.common.SolrDocument
- org.apache.solr.common.SolrDocumentList

- byte[]

- java.util.Iterator
- java.util.Date
- org.apache.solr.common.util.NamedList
- org.apache.solr.common.util.SimpleOrderedMap

- Object[] (items must be of any known type)

How to marshal/unmarshal

```
//example
Map m= new HashMap();
m.put("hello", "world");
OutputStream os = new ByteArrayOutputStream();
new JavaBinCodec().marshal(m,os);

bytes b = ((ByteArrayOutputStream)os).toByteArray()

//now read the Object back
InputStream is =new ByteArrayInputStream(b);
Map m1 = new JavaBinCodec().unmarshal(is);
```

How do I write any custom types ?

Any custom Object has to be first converted to one of the supported types then marshal it.

Versions

As of Solr 4.0, the JavaBin format introduced a new message that is not recognized by earlier version servers, although the javabin version number has not changed. Also, javabin-formatted update messages were previously routed to /update/javabin, but are now sent to /update, and not properly decoded by older servers configured in the usual way. For this reason, you cannot use a Solr 4.x client to send javabin update requests to Solr 3.x and older servers. The easiest workaround for this problem is to use the XML update format, which is what older servers expect at the /update endpoint, or to hold off updating client software until all servers have been migrated to 4.x

As of Solr 3.1, the JavaBin format has changed to version 2.

Version 2 serializes strings differently: instead of writing the number of UTF-16 characters followed by the bytes in Modified UTF-8 it writes the number of UTF-8 bytes followed by the bytes in UTF-8.

When upgrading from Solr 1.4, if you are using the JavaBin format, you will need to ensure that you also upgrade your SolrJ client.

See the SOLR-2034 jira issue for more information.

CategoryQueryResponseWriter