

DataImportHandlerFaq

- I'm using [DataImportHandler](#) with a MySQL database. My table is huge and [DataImportHandler](#) is going out of memory. Why does [DataImportHandler](#) bring everything to memory?
- [DataImportHandler](#) imports a few million documents and then errors, with an error indicating it had lost communication with the database
- I'm using [DataImportHandler](#) with MS SQL Server database with sqljdbc driver. [DataImportHandler](#) is going out of memory. I tried adjusting the batchSize values but they don't seem to make any difference. How do I fix this?
- Is it possible to use core properties inside data-config.xml?
- How would I insert a static value into a field ?
- Why is Transformer specified on entity instead of field ?
- My delta-import goes out of memory . Any workaround ?
- How do I use [DataImportHandler](#) with multiple Solr Cores?
- Running Out of Memory with Postgresql
- How do I use a JNDI DataSource?
- Blob values in my table are added to the Solr document as object strings like B@1f23c5
- Invalid dates (e.g. "0000-00-00") in my MySQL database cause my import to abort
- Why does DIH convert my tinyint(1) columns to boolean values in Solr?

I'm using [DataImportHandler](#) with a MySQL database. My table is huge and [DataImportHandler](#) is going out of memory. Why does [DataImportHandler](#) bring everything to memory?

[DataImportHandler](#) is designed to stream row one-by-one. It passes a fetch size value (default: 500) to `Statement#setFetchSize` which some drivers do not honor. For MySQL, add `batchSize` property to `dataSource` configuration with value -1. This will pass `Integer.MIN_VALUE` to the driver as the fetch size and keep it from going out of memory for large tables.

Should look like:

```
<dataSource type="JdbcDataSource" name="ds-2" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:8889/mysqldatabase" batchSize="-1" user="root" password="root" />
```

[DataImportHandler](#) imports a few million documents and then errors, with an error indicating it had lost communication with the database

This happens frequently with MySQL databases, but it could be a problem with any database.

What's happening here is that Solr (at the Lucene layer) has at least three segment merges scheduled, and one of those merges is very large, taking long enough that the database connection is sitting idle, and either the JDBC driver or the server closes the connection.

Solr uses Lucene's [ConcurrentMergeScheduler](#) by default. The `maxMergeCount` setting for that scheduler controls what happens with merging and indexing. The merge scheduler will always handle the largest merges on the schedule first. As long as the number of merges currently being scheduled is below `maxMergeCount`, ongoing indexing is allowed to run. If the number of merges scheduled reaches that number (default 3), ongoing indexing is paused until the number drops below what's configured. The `maxMergeCount` in the `mergeScheduler` must be increased, so that more merges can be scheduled while still allowing ongoing indexing. This config needs to go at the top level of `solrconfig.xml`. If there is an existing `indexConfig` section, it would need to be incorporated into that section:

```
<indexConfig>
  <mergeScheduler class="org.apache.lucene.index.ConcurrentMergeScheduler">
    <int name="maxThreadCount">1</int>
    <int name="maxMergeCount">6</int>
  </mergeScheduler>
</indexConfig>
```

If the index is on standard spinning disks, leave `maxThreadCount` at 1. If it's on SSD, you can increase it, but probably shouldn't go higher than about 2 or 3.

I'm using [DataImportHandler](#) with MS SQL Server database with sqljdbc driver. [DataImportHandler](#) is going out of memory. I tried adjusting the batchSize values but they don't seem to make any difference. How do I fix this?

There's a connection property called `responseBuffering` in the `sqljdbc` driver whose default value is "full" which causes the entire result set to be fetched. See <http://msdn.microsoft.com/en-us/library/ms378988.aspx> for more details. You can set this property to "adaptive" to keep the driver from getting everything into memory. Connection properties like this can be set as an attribute (`responseBuffering="adaptive"`) in the `dataSource` configuration OR directly in the jdbc url specified in [DataImportHandler](#)'s `dataSource` configuration.

Note that this is not an issue since version 1.2 of the SQL Server JDBC driver. Since version 1.2, "adaptive" is the default setting for `responseBuffering`.

Is it possible to use core properties inside data-config.xml?

If you define the property inside of your `<core/>` in `solr.xml` then you can refer to the property in your `data-config.xml` file directly. However if you define it in the `<solr/>` level, so that it applies to multiple cores, then you need to use the technique below:

- Add your property in the invariant section of `solrconfig's DataImportHandler` element. For example, add this section:
- `<lst name="invariants"> <str name="xmlDataDir">${xmlDataDir}</str> </lst>`
- Use it as `${dataimporter.request.xmlDataDir}` in your `data-config` to access this.
- [see the mail thread](#)

How would I insert a static value into a field ?

add the `TemplateTransformer` to the transformer attribute of the entity and create a field as follows

```
<field column="the_static_field_name" template="any-string-can-be-put-here"/>
```

Why is Transformer specified on entity instead of field ?

A few reasons

- A field declaration is optional because it can be implicit if it is present in the schema
- A Transformer can act on multiple fields . Or it can even do nothing to the data. Say a `LogTransformer` just logs data to a file
- Transformers can be chained . If declared in a field the order of chaining becomes unclear

My delta-import goes out of memory . Any workaround ?

It is possible to do delta import using full-import. Taken from the delta-import example


```
<dataConfig>
  <dataSource driver="org.hsqldb.jdbcDriver" url="jdbc:hsqldb:/temp/example/ex" user="sa" />
  <document>
    <entity name="findDelta" query="select id from item where id in
      (select item_id as id from feature where last_modified > '${dataimporter.
last_index_time}'))
      or id in
      (select item_id as id from item_category where item_id in
      (select id as item_id from category where last_modified > '${dataimporter.
last_index_time}'))
      or last_modified > '${dataimporter.last_index_time}'))
      or last_modified > '${dataimporter.last_index_time}'' rootEntity="false">
    <entity name="item" query="select * from item where id='${findDelta.id}'>
    <entity name="feature" query="select description as features from feature where item_id='${item.
ID}''">
    </entity>
    <entity name="item_category" query="select CATEGORY_ID from item_category where ITEM_ID='${item.
ID}''">
    <entity name="category" query="select description as cat from category where id =
'${item_category.CATEGORY_ID}''">
    </entity>
    </entity>
  </entity>
</document>
</dataConfig>
```

notice that for the entity 'findDelta' `rootEntity` is set to false, so a document is created for each row from 'item'. The command has to be `command=full-import&clean=false`

How do I use `DataImportHandler` with multiple Solr Cores?

If you are using multiple Solr indexes (or multiple cores) in the same Solr webapp, each core will run a separate instance of `DataImportHandler` and each core should have its own `data-config.xml` file in the `conf` directory. See [MultipleIndexes](#) for more details on using multiple indexes.


Running Out of Memory with Postgresql

 **Solr1.4** No more memory issues with this data source configuration:

```
<dataSource type="JdbcDataSource" driver="org.postgresql.Driver" url="jdbc:postgresql://host/db" user="user"
password="password" readOnly="true" autoCommit="false" transactionIsolation="TRANSACTION_READ_COMMITTED"
holdability="CLOSE_CURSORS_AT_COMMIT"/>
```

Using v8.3 of postgres server and JDBC driver.

How do I use a JNDI [DataSource](#)?

 [Solr1.4](#) Setup your JNDI datasource and specify its name in the data-config.xml

```
<dataSource
  jndiName="java:comp/env/jdbc/myDataSource"
  type="JdbcDataSource"
  user=" " password=" " />
```

Blob values in my table are added to the Solr document as object strings like B@1f23c5

The problem occurs because blobs are read as a byte array which does not have a useful toString method. The workaround is to:

1. Use convertType="true" on the JdbcDataSource
2. Write a Transformer which converts the byte[] into a string type
3. You can also use a "cast" sql function which can convert the data type from blob to strings if your database provides such a function

Note that "convertType" attribute uses the target schema field's type name to convert the value returned by the result set. Therefore, any transformers being used on the entity should be aware of the type information.

Invalid dates (e.g. "0000-00-00") in my MySQL database cause my import to abort

Use &zeroDateTimeBehavior=convertToNull parameter in you JDBC connection string to avoid this.

Why does DIH convert my tinyint(1) columns to boolean values in Solr?

This is an oddity of the JDBC specification where tinyint(1) values are treated as bits and their corresponding Java type is a boolean. Use tinyInt1isBit=false parameter in your JDBC connection string to avoid the conversion to boolean value.