Streaming with JDBC

About us

- Gareth Jones
- Co-founder of Bristol Java
- Senior engineer at Brightpearl for 6 years

- @garethsjones84
- gareth.s.jones.84@gmail.com
- https://github.com/garethsjones/spring-petclinic/tree/stream

Brightpearl

- Cloud ERP system
- 11 years old
- Stack:
  - LAMP
  - Java microservices
  - RabbitMQ
  - nodeJS & GoLang
  - Elasticsearch

Talk summary

“How I went about introducing streaming techniques to a legacy codebase”

What it’s not

I have no idea what the latest and greatest Spring libraries are capable of. I have tried several time to see what their Spring Data support of Streams is like but for the life of me I still don’t know.

Maybe they render everything I’ve done obsolete. In fact I hope they have for the sake of anybody starting a greenfield project.

But many of us have to work every day on legacy systems and we have to make do with the libs we have.

Background to the problem

Background to the problem - reports

- Reports, reports, reports, lists & reports
- PHP tier handles display of filters, controls & rows. Makes REST requests to Java service report endpoint.
- Report endpoints accept query params.
- Endpoint returns up to 500 rows at a time + row & search metadata

Background to the problem - exports

- Users want to be able to export large data sets
- PHP tier requests page at a time and processes the data into a choice of terrible formats
- Long-running jobs would time-out
- Even if we could stop the time-outs the work is being done in the wrong place
- What we needed was to move the file generation job into the services (and I’m lazy so if we can reuse the existing SQL and row mapping code that’ll be great)

Spring Pet Clinic - intro

Bad solutions

Fetching data in batches

- Brightpearl’s paginated report endpoint take 2 parameters - page size and index of the first result
- Easy to use and understand, we supply corresponding info in the response metadata
- Easy to code - taking those 2 params and turning them into a SQL LIMIT clause is trivial
- Nice general purpose solution for making queries against the API
- http://api-docs.brightpearl.com/contact/contact/search.html
- contact-search?firstName=Bob&pageSize=10&firstResult=11&sort=contactId|ASC

Fetching data in batches - 2

- General solution != universal solution
- Pitfalls with API pagination are the same the same as query pagination
- On the one hand you have control over batch sizes so you can optimise number or requests vs rows held in memory to your heart’s content
- On the other hand we have the perils of data which won’t sit still.
    - Row insertion & deletion
    - Dependant row updates

Fetching data in one lump

So how about we make a single query and return a Collection of results?

+ It eliminates the problems of frequently updated data
+ You don’t have to execute expensive queries multiple times
- It makes the service explode

Returning a stream from the DAO

- I’m a Java 7 dev at heart so my 1st attempt was to make a ResultSetExtractor which returns Stream<Foo> instead of List<Foo> and then return that stream from the DAO.
- In theory this looks like a pretty easy to use and it wasn’t hard to write.
- Trouble is it doesn’t work
- The way ResultSetExtractor seems to work is that the DB connection is only kept open for the duration of extract()
- So when you pass a Stream out the DAO it’ll be empty when you come to use it. This is not ideal.

Let’s get on with it

A general solution

- If a connection can’t survive outside of the DAO layer then we’re going to have to flip the paradigm on its head and pass the processing code to the DAO instead. Ladies and gentlemen it’s time for lambdas.

Four activities we have to juggle in code

1. Make a query
2. Mapping ResultSet rows to a POJO
3. Operations to use, modify or filter out rows
4. Collecting a return object (or collection)

ResultSetIterator

ResultSetProcessor

The DB connection

- Out ResultSet must be:
    - TYPE_FORWARD_ONLY (this is the default)
    - CONCUR_READ_ONLY (this is the default)
    - A manageable fetch size
- A separate connection pool is recommended