Streaming with BookKeeper

by

Table of contents

1. Summary

When using the BookKeeper API, an application has to split the data to write into entries, each entry being a byte array. This is natural for many applications. For example, when using BookKeeper for write-ahead logging, an application typically wants to write the modifications corresponding to a command or a transaction. Some other applications, however, might not have a natural boundary for entries, and may prefer to write and read streams of bytes. This is exactly the purpose of the stream API we have implemented on top of BookKeeper.

The stream API is implemented in the package `Streaming`, and it contains two main classes: `LedgerOutputStream` and `LedgerInputStream`. The class names are indicative of what they do.

2. Writing a stream of bytes

Class `LedgerOutputStream` implements two constructors and five public methods:

```
public LedgerOutputStream(LedgerHandle lh)
```

where:

- `lh` is a ledger handle for a previously created and open ledger.

```
public LedgerOutputStream(LedgerHandle lh, int size)
```

where:

- `lh` is a ledger handle for a previously created and open ledger.
- `size` is the size of the byte buffer to store written bytes before flushing.

Closing a stream. This call closes the stream by flushing the write buffer.

```
public void close()
```

which has no parameters.

Flushing a stream. This call essentially flushes the write buffer.

```
public synchronized void flush()
```

which has no parameters.

Writing bytes. There are three calls for writing bytes to a stream.

```
public synchronized void write(byte[] b)
```

where:

- `b` is an array of bytes to write.

```
public synchronized void write(byte[] b, int off, int len)
```

where:

- `b` is an array of bytes to write.
- `off` is a buffer offset.
- `len` is the length to write.

```
public synchronized void write(int b)
```

where:

- `b` contains a byte to write. The method writes the least significant byte of the integer four bytes.

3. Reading a stream of bytes

Class `LedgerOutputStream` implements two constructors and four public methods:

```
public LedgerInputStream(LedgerHandle lh) throws BKException,
InterruptedException
```

where:

- `lh` is a ledger handle for a previously created and open ledger.

```
public LedgerInputStream(LedgerHandle lh, int size) throws
BKException, InterruptedException
```

where:

- `lh` is a ledger handle for a previously created and open ledger.
- `size` is the size of the byte buffer to store bytes that the application will eventually read.

Closing. There is one call to close an input stream, but the call is currently empty and the application is responsible for closing the ledger handle.

```
public void close()
```

which has no parameters.

Reading. There are three calls to read from the stream.

```
public synchronized int read() throws IOException
```

which has no parameters.

```
public synchronized int read(byte[] b) throws IOException
```

where:

- `b` is a byte array to write to.

```
public synchronized int read(byte[] b, int off, int len)
throws IOException
```

where:

- `b` is a byte array to write to.
- `off` is an offset for byte array `b`.
- `len` is the length in bytes to write to `b`.