Transactions, MyBatis and Spring

Matt Newman / @mdjnewman

What are transactions?

A transaction comprises a unit of work performed against a database, and treated in a coherent and reliable way independent of other transactions.

http://en.wikipedia.org/wiki/Database_transaction

Why use transactions?

- They're a reliable unit of work
  - Allow correct recovery from failures
    - Keeps database consistent even in cases of system failure, when execution stops
  - To provide isolation between programs accessing a database concurrently

Some theory: ACID

- Atomic
- Consistent
- Isolated
- Durable

Atomic

- From Greek a-tomos, undividable
- In an atomic transaction, a series of database operations either all occur, or nothing occurs.
- Example: ordering a seat on a plane - the system should reserve the sear **and** the payment should be recorded, or neither

Consistent

- The exact meaning of this is more debatable, but it usually means:
  - Any transactions started in the future should ‘see’ all transactions committed in the past
  - Database constraints are not violated (e.g. uniqueness/foreign key constraints)

Isolated

- Unfinished transaction —> No changes visible
- Also refers to whether changes made by other transactions are visible in the current transaction

Durable

- Transactions that are committed should survive permanently

ACID in distributed systems

- … is really hard
- If you're interested in this, read up on CAP theorem and ACID versus BASE for database transactions

http://en.wikipedia.org/wiki/CAP_theorem

http://www.johndcook.com/blog/2009/07/06/brewer-cap-theorem-base/

Why do we care?

Error updating database. Cause: org.postgresql.util.PSQLException: ERROR: ***current transaction is aborted, commands ignored until end of transaction block***

Writing transaction aware code

- Opening/committing/rolling back transactions manually is annoying
- Surely there is some enormous framework that can handle these kinds of things?

Transactions in Spring

```
// the service class that we want to make transactional
@Transactional
public class FooService {

  Foo getFoo(String fooName);

  void insertFoo(Foo foo);

  void updateFoo(Foo foo);
}
```

http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/transaction.html

Configuration required

- MyBatis & Spring integration
  - All services and mappers must be Spring managed beans
    - So no calling `new AccountService()` everywhere!
- Spring DataSourceTransactionManager
- <tx:annotation-driven> element in XML config

http://mybatis.github.io/spring/

Demo!