CS544 EA

# Applications

Transactions

# Spring and Transactions

- We want to add Spring to our applications
  - To make **Spring and Hibernate applications**
  - EMF singleton, ThreadLocal and OpenEMinView are all easy to configure with Spring
  - Real value added is **Transaction Management**

We'll look at Transactions and Spring first.

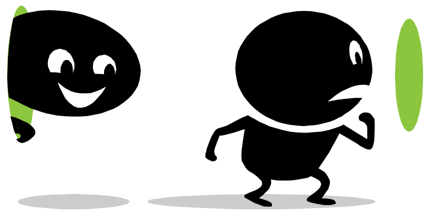
# BMT vs CMT

- Transaction management so far consisted of us writing .getTransaction().begin() and .commit()
  - When using a JEE container this is called Bean Managed Transactions (**BMT**)
  - The container can also manage the transactions for you – Container Managed Transactions (**CMT**)

## Transaction Requirement

- Many developers believe transactions are an optional part of database interactions
- In reality, **there is no such thing as a database interaction without a transaction**

- Most databases default to auto-commit mode
  - Wraps a transaction around each SQL statement
  - Effectively hiding the transaction from view



# Auto Commit Mode

- Auto Commit is good for SQL console work
  - Console work is often ad-hoc (no tx needed)
  - Having to add begin / commit would be more work
- **Auto Commit is bad for applications**
  - More transactions means more overhead
  - Isolation is reduced without transaction boundaries
- Hibernate disables Auto Commit by default
  - Therefore you to specify when to commit! (and begin)

## No Transaction?

- If you don't specify a transaction
  - A transaction will still be open at the JDBC level
  - Hibernate has turned off auto-commit
  - Hibernate will throw **IllegalStateException**

```
Exception in thread "main" java.lang.IllegalStateException: Transaction not successfully started
	at org.hibernate.engine.transaction.internal.TransactionImpl.commit(TransactionImpl.java:98)
	at cs544.hibernate01.basic.App.main(App.java:21)
```