Concurrency Control

* No concurrency control
  + Bad choice
  + Not as rare as it should be
  + Last one to save, “wins”
    - Earlier saves by other people can silently disappear - yikes!
  + Why? Because it’s the easiest
* Pessimistic Concurrency Control
  + Expect collisions are not rare
  + Lock a record for update
    - In some cases lock can expire (server-side code)
      * If lock has expired, save fails
    - If lock hasn’t expired (or never expires)
      * Save always succeeds
    - Need to decide is there should be any read access to this locked record
    - Need an admin function to forcibly unlock an abandoned lock
    - When lock request succeeds, make sure the freshest data is sent to the UI
* Optimistic Concurrency Control
  + Collisions are expected to be rare
  + Keep a ‘revision number’ with object
  + When a Save occurs, check first that the revision number matches, then increment number (and save changes)
    - If it doesn’t match, throw ConcurrentModifictionException (throw away changes)