**Backendless Part 2 – Overwrite an Existing Record**

The previous example shows how to setup Backendless in your app, save data, and retrieve data. Complete that first before doing this example.

**Saving Over a Record**

Backendless can save over an existing record instead of creating a new one. To do that, save an object whose “objectId” field matches an existing record’s objectId.

This objectId field is automatically created by backendless. You can add this field to your Comment class (capitalization must match exactly) and create getters and setters to access it.

To get the objectId of an existing record, you have to connect to Backendless and fetch that info from your Comment table. Here is one way to do that:

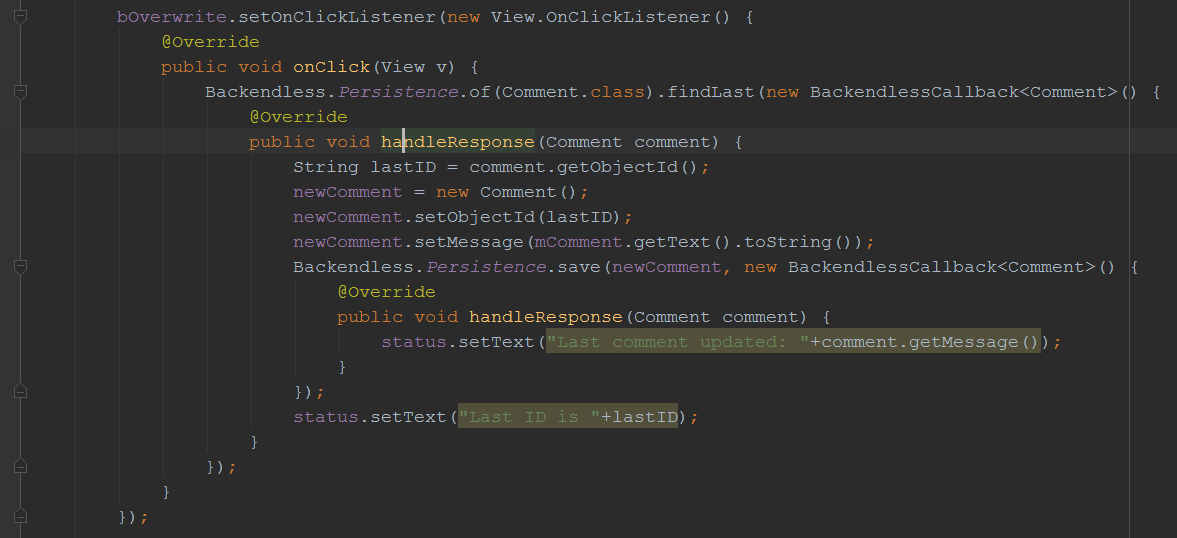
1. Use the Persistence.findLast() method to retrieve the most recent Comment object
2. Use that objects getObjectId() method to get the matching objectId
3. Create a new comment and use its setObjectId() method to give it the String from ii)
4. Use Persistence.save() to save this new comment. It will save over the previous object with that objectID.

**Timing Considerations**

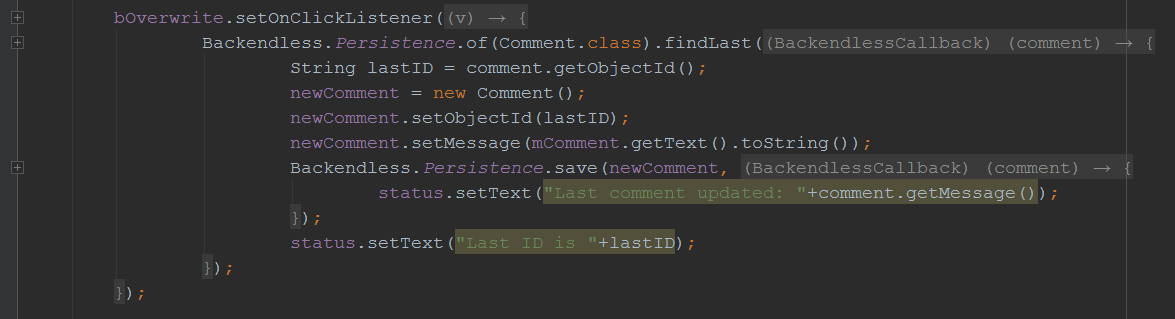
You don’t want your app to save the record until it has the objectId back from the cloud. You care about the order that these operations execute. If you start the findLast() method, then start Persistence.save() method, you aren’t able to control which of those will complete first. This problem is called a “race condition.”

To make your program wait for the objectId, you can put the Persistence.save() call in the handleResponse() method nested in your Persistence.findLast() call. This forces the app to wait until objectId comes back before it starts to save the new comment. Look at the example app’s Overwrite button onClick function to see what this nested structure looks like.

Tip: you can hide or expand some parts of your code to make it easier to read using the [-] and [+] buttons on the left edge of the editor pane. The examples below show the “Overwrite most recent” button’s onClickListener code fully expanded and fully compacted:



*Fully expanded onClickListener with two nested BackendlessCallbacks fully displayed*



*Compacted view of the same code. The arrows indicated where code is hidden.*

**Queries**

If you want to replace a specific record instead of simply replacing the most recent one, you can create a query to find the one(s) you want. That is how CollegeApp gets the objectId of the Profile associated with a certain email address. Query examples will be explained in a future activity.