**Backendless Help Guides – Paging**

**Example app**: Backendless\_Overwrite.zip

**Purposes of this example:**

* Start to become familiar with BackendlessCollection objects
* Learn to look at all of the records in a Backendless data table instead of only the first 10

When you retrieve data with Backendless using Backendless.Persist.find(), you only get 10 results at a time with the default settings. Those 10 items are a “page” of results. It limits the number of results you get at a time so that the server doesn’t get overwhelmed by requests for millions of records at a time, and your app can run faster if it doesn’t have to deal with such large amounts of data.

A BackendlessCollection object has the ability to request the next page of data objects using the nextPage() method. It also has a previousPage() method. These methods both need to be run in the background just like a find() method. That is done by creating a BackendlessCallback instance as an argument.

If you look at the app, one thing you might notice is that there is now a global BackendlessCollection object called “commentsCollection.” The code is organized in this way so that you can save the latest response from the database to this object and then run nextPage() or previousPage() on it.

As a reminder, when you request data from Backendless, the data becomes available inside the handleResponse() method of aBackendlessCallback instance. Getting the data out of that method and using it later can be tricky. Storing it in a locally declared object (commentsCollection, in our case) is one way around this problem.

The BackendlessCollection object includes the List<> of Comment objects, available using getData(). It also has other things, such as the ability to report the total number of objects in the online table, getTotalObjects(). It can also return the query object that is used to find the objects with getQuery().

A “query” is a request for information from a database. Backendless uses BackendlessDataQuery objects to define these queries. You can set the number of results that you want per page using setPageSize(), and the “offset,” which is the starting record for this query. For example, if a table has 25 records and you specify an offset of 10 and a page size of 5, you would get record numbers 10 through 14 back in your response. The default offset is zero, and default page size is 10 if you don’t specify those values.

A query can also specify a “WHERE clause” that can be used to filter certain records from a data table. That will be explored in a future example. For now, this example app only uses queries to help with data paging.

Look through the data to see how it is able to request next page and previous page. Try to understand how it is getting the total number of records and display which records are being shown (“Showing records 6 to 10 of 35”).

This example app is organized a bit differently than the other ones, because it was starting to repeat some code. Each time it displays a page of records, it uses the displayRecords() method, which is defined in this app. This method takes a List<Comment> object as its input argument. Using that information, it displays the message from each comment in the status TextView. Then it looks at the commentsCollection object (a BackendlessCollection) and uses it to display the records numbers that are being shown in the tvShowingRecords TextView.

When the app first loads in onCreate(), a new BackendlessDataQuery is created. You can change the number of records per page to change the way the app displays records (try this!). Then it loads the first page of results as the Activity loads. Once this is set, the BackendlessCollection will keep using this value for subsequent queries when changing pages.