

Working with Parse.com

[Parse.com](#) is a popular backend-as-a-service ([BaaS](#)) that essentially gives developers a virtual database and app server on the cloud. You can create your own classes with a unique name that accepts any number of parameters and various data types. These classes can then be invoked by the mobile SDK, and we get the response from the cloud server. This response can consist of any number of data objects. For standard database queries, this SDK comes with an additional powerful feature: client-side caching.

I've been working on Parse.com and I've used it as a backend, allowing for syncing data between app and website/cms. For the most part, it has been an excellent experience, and I'd recommend it to anyone needing a hosted backend solution for a web, desktop or mobile application.

Parse SDK is available for all major [platforms](#) and there are various [services](#) provided by the parse. I'm going to explain for the iOS platform.

How to get started:-

- Create free account on parse.com.
- Create your project.
- Download and include the SDK in your project or download the blank project.
- Include the Application Id and Client Key generated by parse into your project.
- Import <Parse/Parse.h>
- Start coding

How does Parse work:-

Storing data on Parse is built around the [PFObject](#). Each PFObject contains key-value pairs of JSON-compatible data. This data is schema less, which means that you don't need to specify ahead of time what keys exist on each PFObject. You simply set whatever key-value pairs you want, and the parse backend will store it..

Each PFObject has a class name that you can use to distinguish different sorts of data. Whenever any data is added, an unique objectId is created by the Parse server.

Saving Objects:-

```

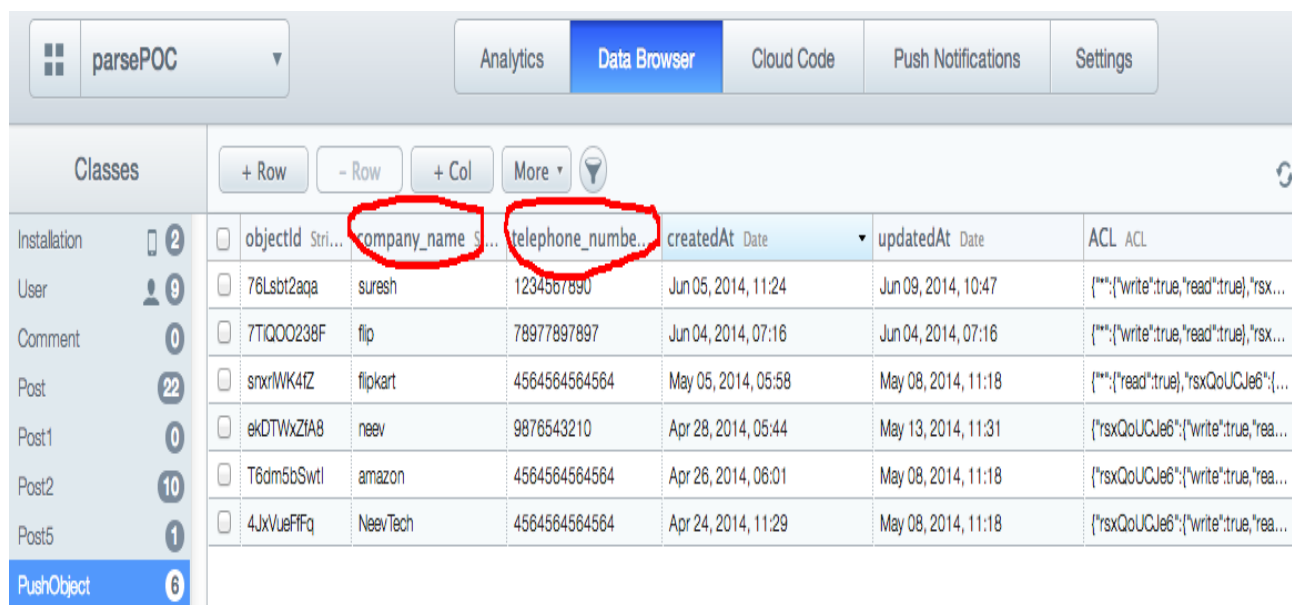
PFObject *userInfo = [PFObject objectWithClassName:@"PushObject"];
userInfo[@"company_name"] = @"your_company_name";
userInfo[@"telephone_number"] = @"your_telephone_number";
[userInfo saveInBackground];

```

SaveInBackground method will save the data your_company_name and your_telephone_number to the parse server.

Note: You can use the saveInBackgroundWithBlock or saveInBackgroundWithTarget:selector: methods to provide additional logic which will run after the save completes.

Image showing the saved objects in Data Browser.



Classes	+ Row	- Row	+ Col	More			
Installation 2	<input type="checkbox"/>	objectid	company_name	telephone_number	createdAt	updatedAt	ACL
User 9	<input type="checkbox"/>	76Lsb2aga	suresh	1234567890	Jun 05, 2014, 11:24	Jun 09, 2014, 10:47	{**:{write:true,read:true},rsx...
Comment 0	<input type="checkbox"/>	7TQOO238F	flip	78977897897	Jun 04, 2014, 07:16	Jun 04, 2014, 07:16	{**:{write:true,read:true},rsx...
Post 22	<input type="checkbox"/>	snxrWK4fZ	flipkart	4564564564564	May 05, 2014, 05:58	May 08, 2014, 11:18	{**:{read:true},rsxQoUCJe6:{...
Post1 0	<input type="checkbox"/>	ekDTWxZfA8	neev	9876543210	Apr 28, 2014, 05:44	May 13, 2014, 11:31	{rsxQoUCJe6:{write:true,rea...
Post2 10	<input type="checkbox"/>	T6dm5bSwtl	amazon	4564564564564	Apr 26, 2014, 06:01	May 08, 2014, 11:18	{rsxQoUCJe6:{write:true,rea...
Post5 1	<input type="checkbox"/>	4JxVueFFq	NeevTech	4564564564564	Apr 24, 2014, 11:29	May 08, 2014, 11:18	{rsxQoUCJe6:{write:true,rea...
PushObject 6							

Fig-1.

Retrieving Objects :-

```

PFQuery *query = [PFQuery queryWithClassName:@"PushObject"];
[query findObjectsInBackgroundWithId:@"objectId" block:^(PFObject
*userInfo, NSError *error) {

```

```

// Do something with the returned PFObject in the gameScore variable.
NSLog(@"%@@", userInfo);
}];

```

To get the values out of the PFObject, you can use either the objectForKey: method or the [] subscripting operator:

```
NSString *company name = userInfo[@"company_name"];
```

Relational Data :-

Objects can have relationships with other objects. To model this behavior, any PFObject can be used as a value in other PFObjects. Internally, the Parse framework will store the referred-to object in just one place, to maintain consistency.

```
// Create a new Post object and create relationship with PFUser
```

```
PFObject *newPost = [PFObject objectWithClassName:@"Post"];
```

```
[newPost setObject:[textView text] forKey:@"textContent"];
```

```
[newPost setObject:[PFUser currentUser] forKey:@"author"]; // One-to-Many relationship created here!
```

```
// Save new Post object in Parse
```

```
[newPost saveInBackgroundWithBlock:^(BOOL succeeded, NSError *error)
{
    if (!error) {
        //required logic goes here..
    }
}];
```

In the above code whenever a user login the objectId of that particular user is added into the author column of the Post class to keep a track of the user who has added some content into the textContent column of the Post class. In this way we are creating one to many relation. Similarly many to many relation can also be created.

Classes	+ Row	- Row	+ Col	More ▾		
Installation 2	<input type="checkbox"/>	objectId String	author Pointer<User>	likes Array	textContent String	createdAt Date ▾ updatedAt Date
User 9	<input type="checkbox"/>	mBECmAyjIN	rsxQoUCJe6	["rsxQoUCJe6"]	dfgfgfg	Jun 09, 2014, 11:12 Jun 09, 2014, 11:...
Comment 0	<input type="checkbox"/>	OyJN2uldtd	rsxQoUCJe6	["rsxQoUCJe6"]	suresh	Jun 05, 2014, 11:29 Jun 09, 2014, 07:...
Post 22	<input type="checkbox"/>	SYHJXOIqH	rsxQoUCJe6	(undefined)	ererererer	May 20, 2014, 07:04 May 20, 2014, 0...
Post1 0	<input type="checkbox"/>	jk2pk6Ctgn	wA1g9C7fY5	["wA1g9C7fY5"]	After adding one more content	May 20, 2014, 05:18 May 20, 2014, 0...
Post2 10	<input type="checkbox"/>	jhHTtO6hFK	wA1g9C7fY5	(undefined)	Sandy account	May 20, 2014, 05:17 May 20, 2014, 0...

Fig-2.

Push Notification:-

Push Notifications are a great way to keep your users engaged and informed about your app. You can reach your entire user base quickly and effectively.

We will begin on the Apple Developer website to create an SSL certificate associated to an App ID and a provisioning profile [here](#). Once the provisioning profile and the certificate is created it is added to the Push Notification tab of the dashboard and we are ready to send the notification.

Parse have provided various ways of sending Push Notification, these are-

- From Parse Website
- From Rest API
- From the App
- From Cloud Code

Analytics:-

Parse provides a number of hooks for us to get a glimpse into the ticking heart of our app. without having to implement any client-side logic, we can view real-time graphs and breakdowns (by device type, Parse class name, or REST verb) of our API Requests in our app's dashboard and save these graph filters to quickly access just the data we're interested in.

App-Open / Push Analytics:

By adding the following line we'll begin to collect data on when and how often your application is opened.

```
[PFAalytics trackAppOpenedWithLaunchOptions:launchOptions];
```

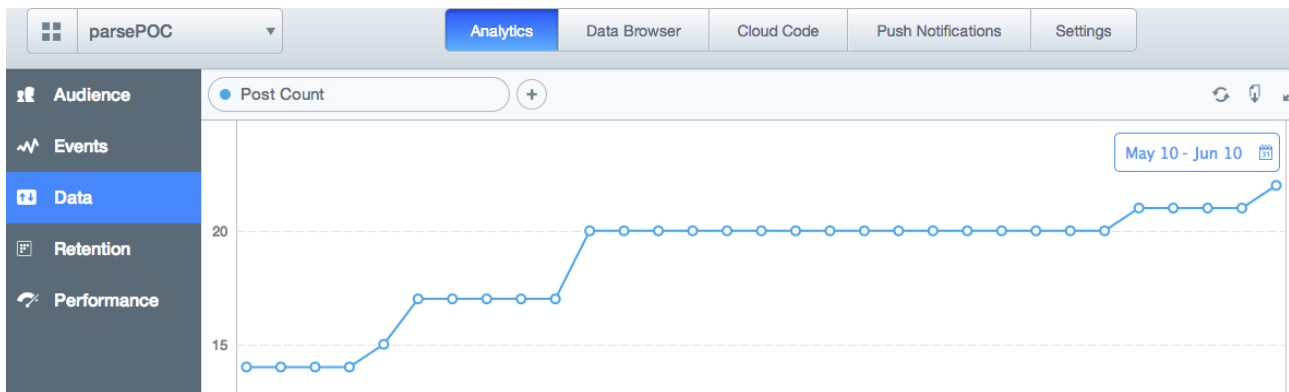
Custom Analytics :

PFAalytics also allows us to track free-form events, with a handful of NSString keys and values. These extra dimensions allow segmentation of our custom events via our app's Dashboard.

Example:

Say our app offers search functionality for any listings, and you want to track how often the feature is used, with some additional metadata.

```
NSMutableDictionary *dimensions = @{
    // Define ranges to bucket data points into meaningful segments
    @"priceRange": @"1000-1500",
    // Did the user filter the query?
    @"source": @"craigslist",
    // Do searches happen more often on weekdays or weekends?
    @"dayType": @"weekday"
};
// Send the dimensions to Parse along with the 'search' event
[PFAalytics trackEvent:@"search" dimensions:dimensions];
```



Above figure-3 is showing the analytics for the data posted in the Post class on day to day basis from May 10 to jun 10.

Cloud Code:-

Parse has provided the option of writing our own logic. For complex apps, sometimes we just need a bit of logic that isn't running on a mobile device. Cloud code makes this possible. The only difference is that this code runs in the Parse Cloud rather than running on a mobile device. When we update our Cloud Code, it becomes available to all mobile environments instantly.



Fig-4.

In the above figure-4 whenever the data(`telephone_number`) of `PushObject` (shown in fig-1) is updated an alert is sent to the device using push notification saying New Number Updated: “new number”.

Parse Security:-

Connection Between Client and Server :

All connections are made with HTTPS and SSL, and Parse will reject all non-HTTPS connections.

Class-Level Permissions :

Restricting class creation from your app.

Configuring class-level permissions on your schema.

Creating restrictions on how your data is accessed by other users.

Object-Level Access Control :

The easiest way to control who can access data is through access control lists, commonly known as ACLs.

Conclusions:

Parse is a great product and it represent a huge boon for developers who want to focus on the app they're building and not the infrastructure that supports it.