# Push Notifications with Parse

**PART ONE**

**Author: Muhammad Shahab Hameed**

## Sources

Project homepage on Github - <https://github.com/shahabhameed/parse_push_notifications>

Download Zip – <https://github.com/shahabhameed/parse_push_notifications/archive/master.zip>

## Overview

This is a three-part tutorial series that will demonstrate how to send Parse’s Push Notifications from the backend server to the client devices (Android & IPhone). In the first part, a REST based backend server will be built with Java EE Spring4 and Jersey framework that will utilize Parse’s Rest API’s to send Push Notifications. In the second part, an Android client will be built using Android Studio IDE targeting Android Honeycomb (API SDK Level 11) and Marshmallow (API SDK Level 23) that will receive push notifications send by the backend server. Finally, in the third part, an IPhone client will be built using Xcode (7.0.1) for receiving push notifications.

## Requirements

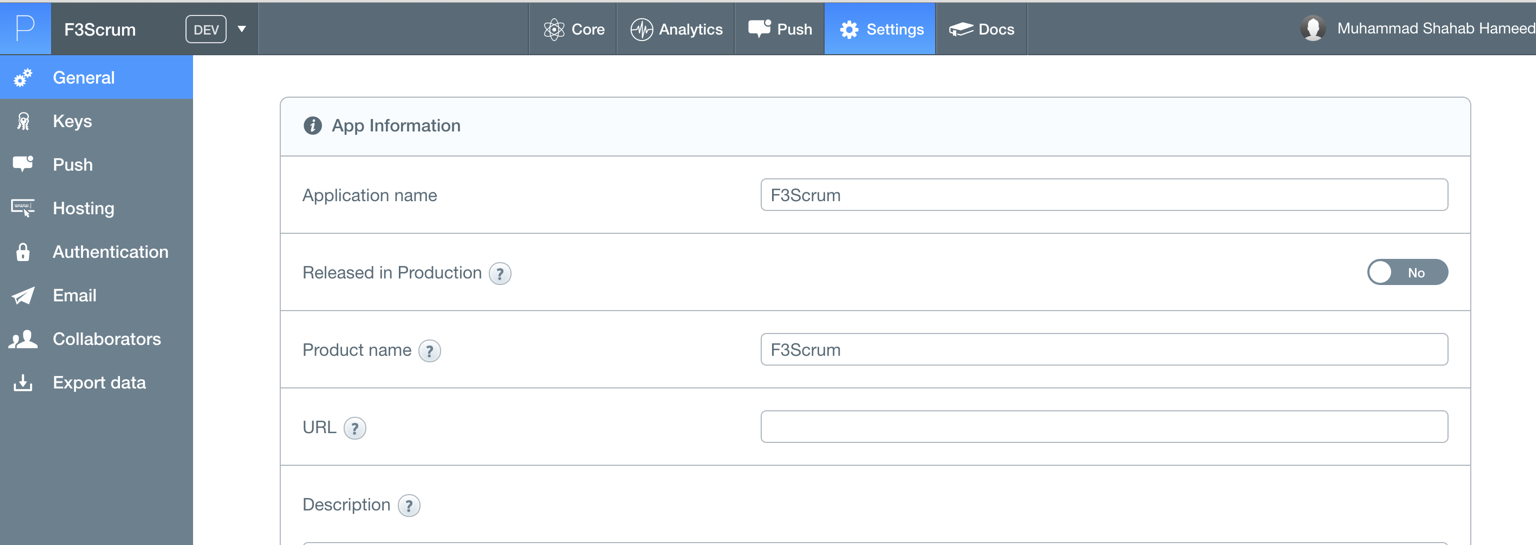
The following software have been used to implement this tutorial.

1. Java SDK 1.8 (Version 8 update 65)
2. Eclipse Java EE IDE (Version Mars, Release 4.5.1)
3. Tomcat 8 (Version 8.028)

## Implementing Push Notifications using Parse REST API

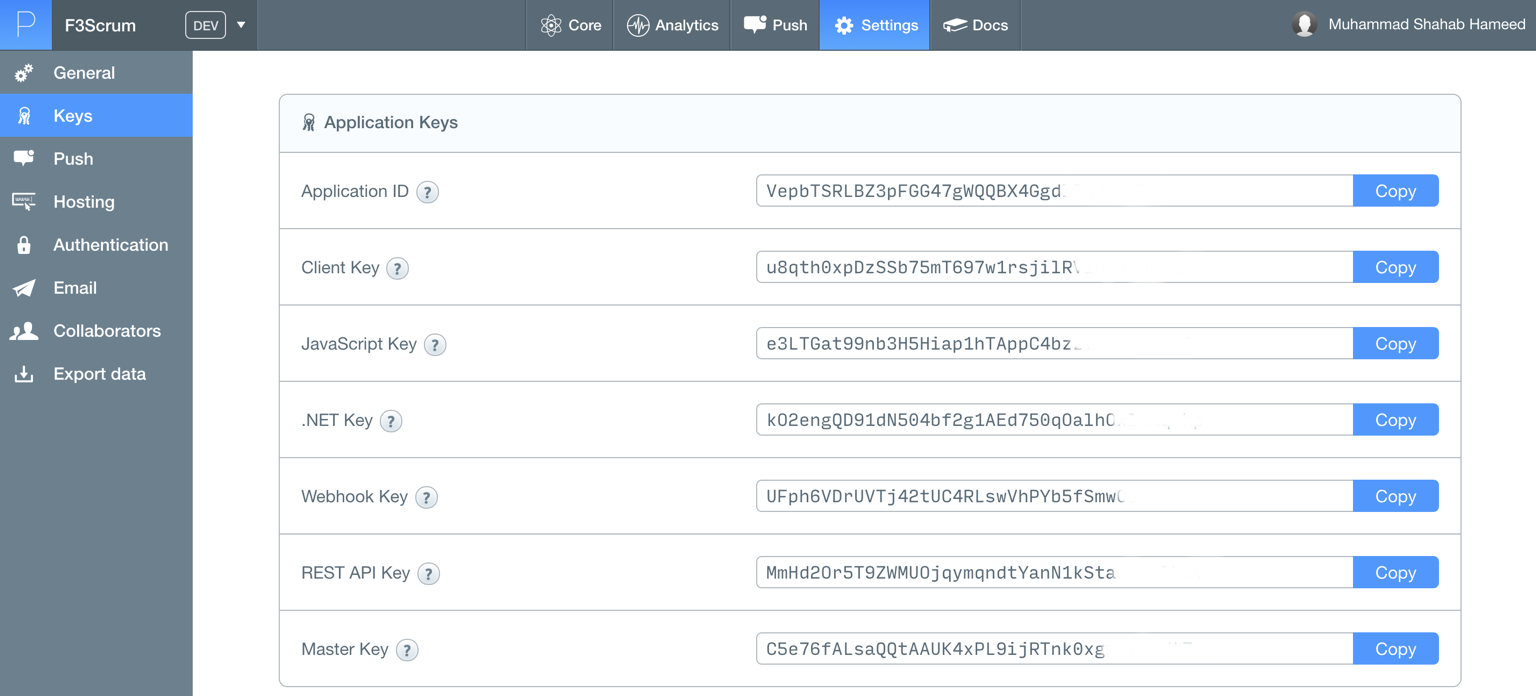
Step 1

First, we are going to register an application on Parse. For that navigate to [www.parse.com](http://www.parse.com) and create an account. I have signed up and created an application “F3Scrum” as shown below.



Once the application is created go to “Settings” -> “Keys” and note down the following keys that will be used later as configuration in our backend server implementation.

1. Application ID
2. Client Key
3. Rest API Key
4. Master Key

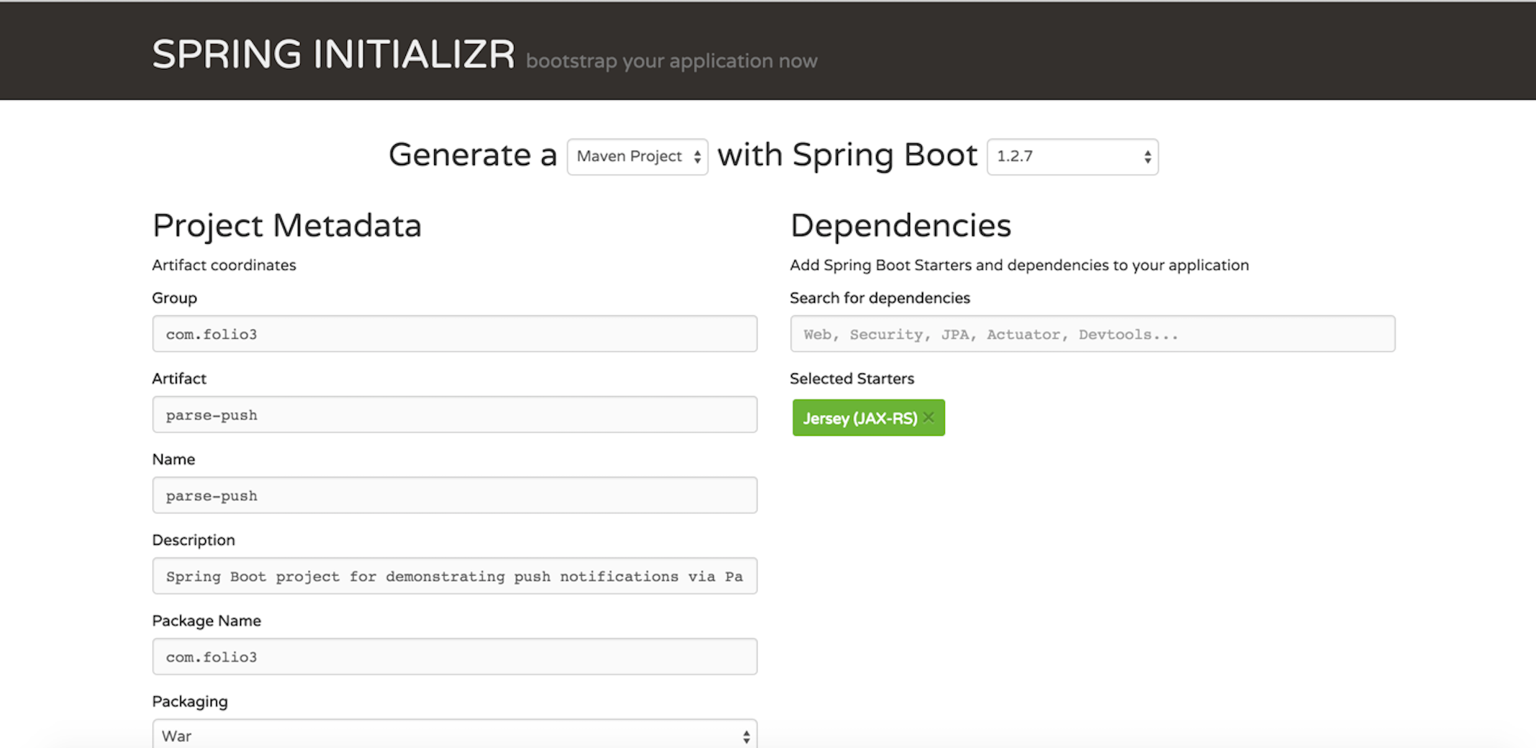


Step 2

Next, we are going to bootstrap our JavaEE Spring application using Spring Boot. Spring Boot facilitates in creating a skeleton for our application using a build dependency system.

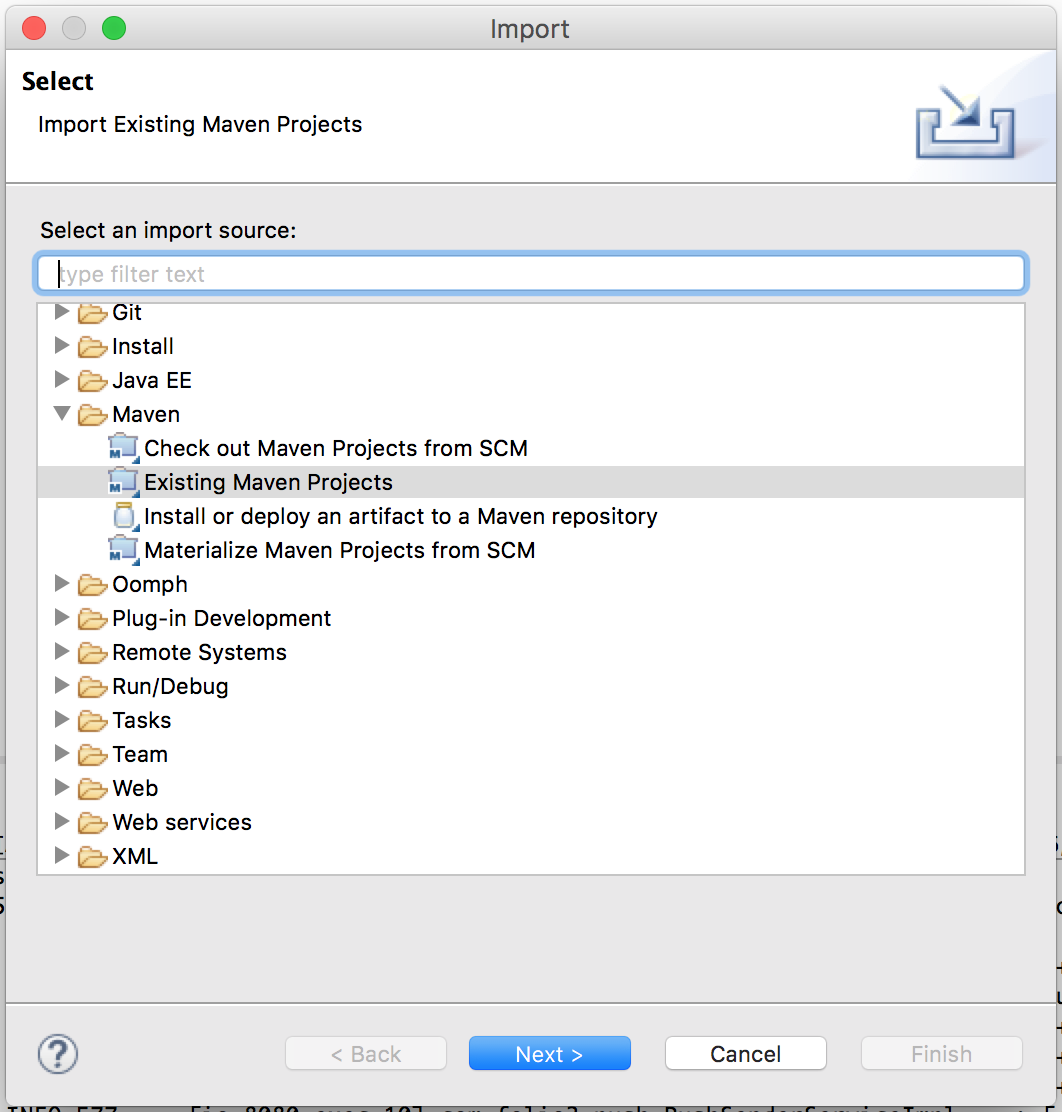
For this, we go to Spring Initializr website ([www.start.spring.io)](http://www.start.spring.io)) and configure our application. Since we are using Jersey REST framework for developing APIs we will select Jersey (JAX-RS) from the options provided. The packaging is done in web-archive (war) format so that it could be deployed on our Tomcat application server. Also we are going to select Maven as our build dependency system.

Once all the fields have been filled, we will click “Generate Project” and download it on our computer.



Step 3

Once the project has been downloaded, we will extract and import it in Eclipse IDE. Click “File” -> “Import…” -> “Existing Maven Projects” and browse the extracted folder and download it. The maven will auto download all the dependencies required for building this project defined in the “pom.xml” file.



Step 4

Next, we need to modify “pom.xml” file to include Google’s Gson library that maps Java class to JSON and IOUtilities package as one of the “**dependencies”** in our project.

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.4</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

</dependency>

