EASY LEADERBOARD (Facebook + PlayFab)

Instruction manual, version 2.0.0

Changelog

- New Sample Scene (more decoupled, optimized and mobile-friendly, totally redesigned from scratch);
- Invite functionality removed (App Invites are deprecated by Facebook).

The purpose of Easy Leaderboard is to simplify the implementation of leaderboards in your game, utilizing two services that are much used by developers: Facebook and PlayFab.

Why using Easy Leaderboard?

- Quick implementation: create a worldwide leaderboard in your game in a matter of minutes;

- Picture: shows the profile picture of the players;

- Incremental search: optimized logic for retrieving PlayFab records in an incremental way;

- Flexible search: you define how many records are supposed to be retrieved on each increment;

- Unlimited statistics: handle as many PlayFab statistics as needed;

- UI-independent: the script has no dependency on UI components, which gives you extra flexibility and ease to adapt it to your game;

- Earn more players: spread your game with Facebook Share!

1. PROJECT REQUIREMENTS

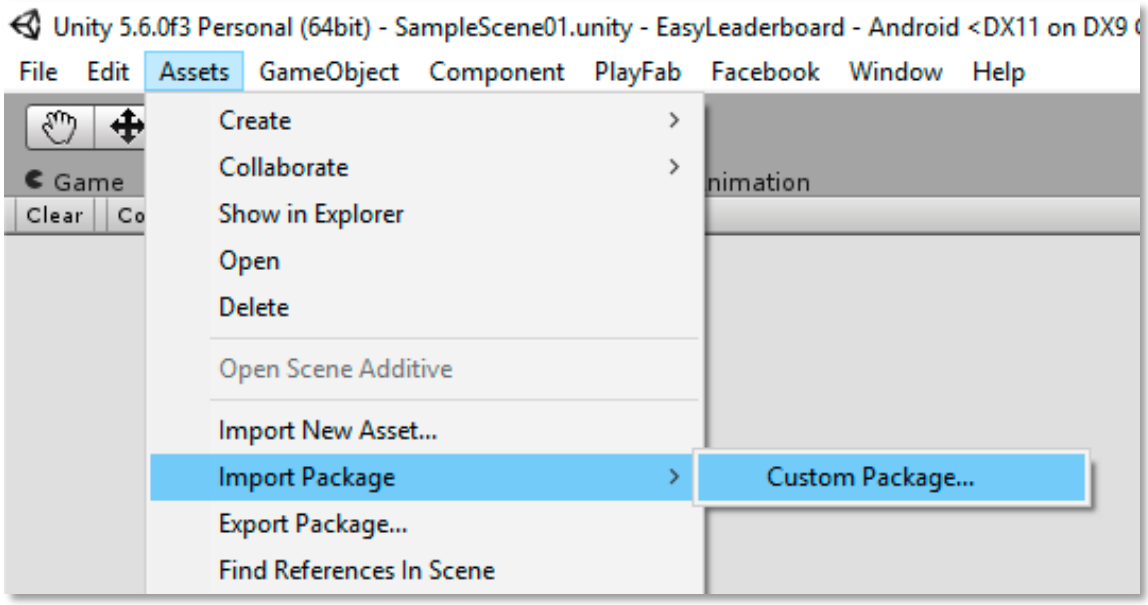
A lot of errors will appear as you import Easy Leaderboard on your project. It occurs because Easy Leaderboard depends on classes that are made available by Facebook and PlayFab by their respective development kits (SDKs) and those kits are not included on the project. In this section we will see not only how to fix those errors but also how to get your app properly configured and ready to use on Facebook and PlayFab.

1.1 Facebook

1.1.1 Importing the SDK

a) Access this link https://developers.facebook.com/docs/unity/downloads and download Facebook SDK.
ATTENTION: this Asset was developed and tested using Facebook SDK version *7.15.0* so we highly recommend you to use this version to prevent unknown errors and/or failures.

b) In Unity, click on Assets > Import Package > Custom Package...



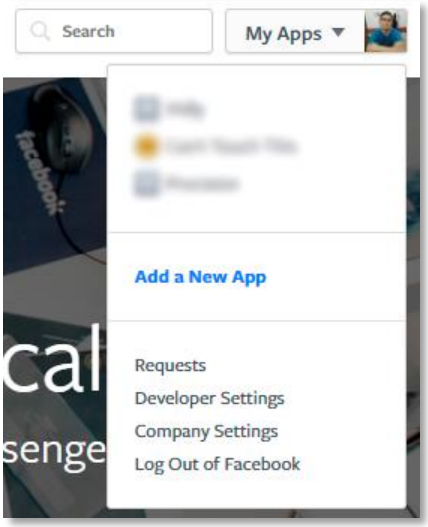
c) Search for the *.unitypackage* file inside the package you've just downloaded and click on Import (usually on FacebookSDK directory).

1.1.2 Facebook developer account

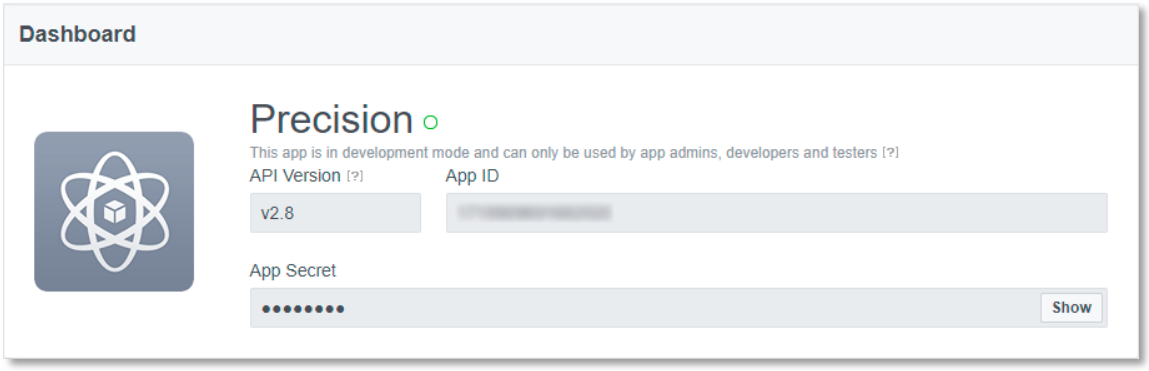
If you don't have a Facebook developer account, create an account following the instructions provided by this page: https://developers.facebook.com.

1.1.3 App configuration on Facebook

a) In the upper right corner click on *Add a New App*:



b) A screen like this one will be displayed:

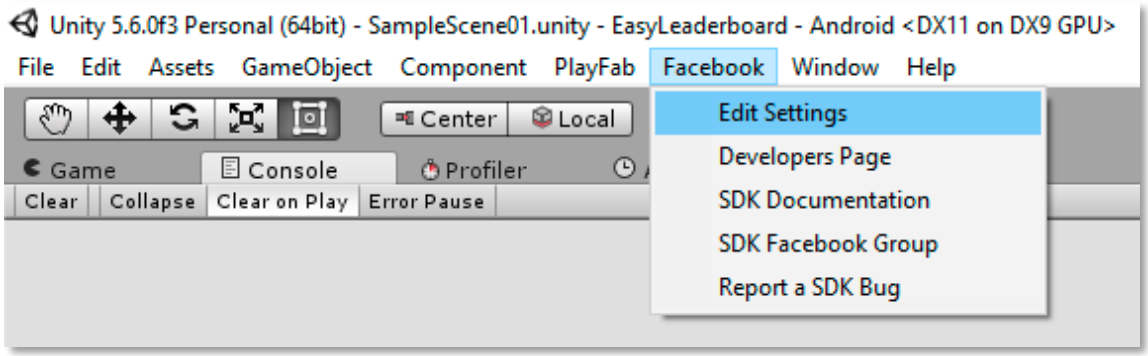


c) Copy the content of the *App Id* field (it will be used later).

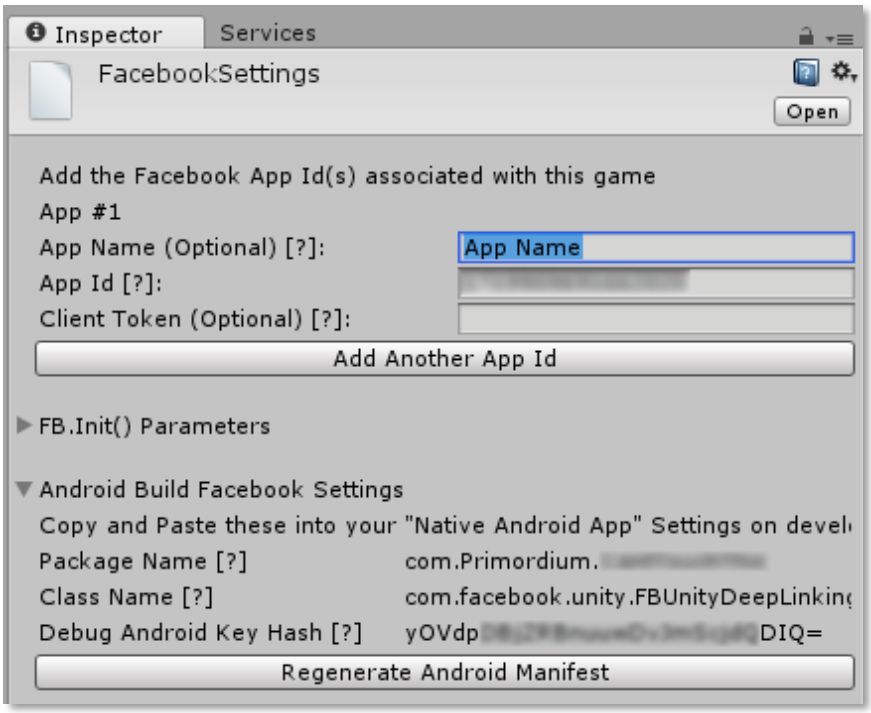
d) In the upper left corner click on Settings. Then click on *Add platform* (Android will be used on this example).

e) To fill in the next fields (*Google Play Package Name, Class Name* and *Key Hashes*) we need to get back to Unity.

f) Click on Facebook > Edit Settings (restart Unity if this option is not being displayed).



g) After clicking this option the following settings must be displayed in Unity Editor:



h) Fill in the *App Name* field with your application name and paste the App Id (the info we've copied earlier) on the *App Id* field.

i) Now copy the content of the next 3 fields, paste on the 3 respective fields on Facebook developer console and click on *Save Changes*.
PS: the first time you test your app on a device will be displayed a message saying the Key Hash is not valid. Type the Key Hash, add it to the Key Hashes field and save changes. Then Facebook will be able to identify your app and everything will work fine.

1.2 PlayFab

1.2.1 Importing the SDK

a) Access this link https://api.playfab.com/sdks/unity and download PlayFab SDK.

b) Repeat the step 1.1.1b but now importing the PlayFab SDK.

1.2.2 PlayFab developer account

If you don't have a PlayFab developer account, create an account following the instructions provided by this page: https://developer.playfab.com.

1.2.3 App configuration on PlayFab

a) To create a new app on PlayFab click on New Game, complete your app information and save changes.

b) On the left, click on Settings and then click on API tab.
c) Copy the *Title Id*, checkmark the *Allow client to post player statistics* and save changes. This option allows your app to handle player statistics.

2. SETTING UP EASY LEADERBOARD

2.1 Recommended use

a) Drag the EasyLeaderboardManager prefab and drop on the hierarchy of the first scene of your game. The script attached to the prefab (FacebookAndPlayFabManager) is implemented with the Singleton Pattern and uses DontDestroyOnLoad method from MonoBehaviour class. This means that as soon as it is loaded you can use every feature this script provides from anywhere in your code regardless of the scene.
b) Fill in the *PlayFab Title Id* field with the *Title ID* you copied from PlayFab developer console.
c) The fields from *Facebook Share Info* section are only required if you intend to use the Facebook Share feature.

3. USING EASY LEADERBOARD

3.1 Public methods

Public methods of FacebookAndPlayFabManager class has void return type and can be called from anywhere in your code.

3.1.1 LogOnFacebook: logs the player into Facebook.

Parameters:

- Action<ILoginResult> *successCallback* (optional): action to be executed when the process is done correctly.

- Action<ILoginResult> *errorCallback* (optional): action to be executed when the process fails.

PS: When the process is done correctly this method will be responsible not only to execute the action passed on *successCallback* parameter but also to log the player into PlayFab and set values to the following properties of the class:

- bool *IsLoggedOnFacebook*: is the player logged on Facebook?

- bool *IsLoggedOnPlayFab*: is the player logged on PlayFab?

- string *FacebookUserName*: player's Facebook profile name.

- Sprite *FacebookUserPictureSprite*: player's Facebook profile picture.

Those properties are public and read-only so you can read their values from anywhere in your code as it is with the public methods.

3.1.2 LogOutFacebook: logs the player out of Facebook.

3.1.3 GetFacebookUserName: gets the player's Facebook profile name.

Parameters:

- string *id*: unique identifier of a Facebook profile.

- Action<IGraphResult> *successCallback* (optional): action to be executed when the process is done correctly.

- Action<IGraphResult> *errorCallback* (optional): action to be executed when the process fails.

3.1.4 GetFacebookUserPicture: gets the player's Facebook profile picture.

Parameters:

- string *id*: unique identifier of a Facebook profile.

- int *width*: width the returning image is supposed to have.

- int *height*: height the returning image is supposed to have.

- Action<IGraphResult> *successCallback* (optional): action to be executed when the process is done correctly.

- Action<IGraphResult> *errorCallback* (optional): action to be executed when the process fails.

Some recent Unity versions are not compatible with the default way of getting the player's Facebook profile picture. Use the following method If you are having trouble getting the profile picture.

3.1.4 (Alternative) GetFacebookUserPictureFromUrl: alternative way to get the player's Facebook profile picture.

Parameters:

- string *id*: unique identifier of a Facebook profile.

- int *width*: width the returning image is supposed to have.

- int *height*: height the returning image is supposed to have.

- Action<IGraphResult> *successCallback* (optional): action to be executed when the process is done correctly.

- Action<IGraphResult> *errorCallback* (optional): action to be executed when the process fails.

3.1.5 GetLeaderboard: gets values from a given PlayFab statistic.

Parameters:

- string *statisticName*: name of the PlayFab statistic to be retrieved.

- bool *friendsOnly*: only retrieve info from Facebook friends?

- int *maxResultsCount*: maximum number of records to retrieve.

- Action<GetLeaderboardResult> *successCallback*: action to be executed when the process is done correctly.

- int *startPosition* (optional, default value = 0): starting point to retrieve the statistic values.

3.1.6 UpdateStat: updates the value of a given PlayFab statistic.

Parameters:

- string *statisticName*: name of the PlayFab statistic to be updated.

- int *value*: value the statistic must receive.

- Action<UpdatePlayerStatisticsResult> *successCallback* (optional): action to be executed when the process is done correctly.

PS: it is recommended to create constants with the names of the statistics you will need to handle to avoid typos (look at the PlayFabStatConstants class at EasyLeaderboard > Scripts > Utility).

3.1.7 ShareOnFacebook: shares a post on Facebook.

Parameters:

- Action<IShareResult> *successCallback* (optional): action to be executed when the process is done correctly.

- Action<IShareResult> *errorCallback* (optional): action to be executed when the process fails.

PS: fill in the fields *Content Url, Content Title, Content Description* and *Picture Url*.

4. CONTACT

If you have any doubt you can get in touch with our support sending an e-mail to primordiumlabs@gmail.com (please set the subject as Easy Leaderboard). Your message will be replied as soon as possible!

Please encourage us to improve our asset giving your opinion about Easy Leaderboard on Unity Asset Store.

Thank you!

PRIMORDIUM
GAME STUDIO