

[Firebase] Unity3d 유저 인증

Programming/Firebase / 선비리즘 / 2019. 12. 17. 16:41

목차

Tutorial

이름	버전 (2019-12-04 기준)
Unity3d	2018.4.13f1 (LTS)
Firebase Unity SDK	6.7.0

Unity3d 클라이언트에서 Firebase Unity SDK 만을 이용하여 Firebase 인증을 사용한다.

Firebase Setting

이 글은 이메일/비밀번호, 구글, 익명 인증만 설명합니다.

1. 아래 링크에서 새로운 Firebase 프로젝트를 만들거나 기존의 Firebase 프로젝트를 사용한다.

<https://seonbicode.tistory.com/28>

2. 새로운 앱을 등록한다.

인증은 Unity3d 앱에서만 처리할 예정이므로 **안드로이드 앱** 관련 내용만 참고하자.

https://seonbicode.tistory.com/43#Firebase_Project_New_App_Add

3. 아래 링크에서 **인증** 항목을 참고하여 Firebase 세팅을 한다.

<https://seonbicode.tistory.com/43#Authentication>

4. Keystore 와 Firebase 에서 SHA 인증서 지문 세팅을 한다.

https://seonbicode.tistory.com/43#SHA_%EC%9D%B8%EC%A6%9D%EC%84%9C_%EC%A7%80%EB%AC%B8

5. SHA 인증서 지문 세팅까지 끝낸 후 아래 링크에서 google-services.json 관련 세팅을 한다.

<https://seonbicode.tistory.com/43#google-services.json>

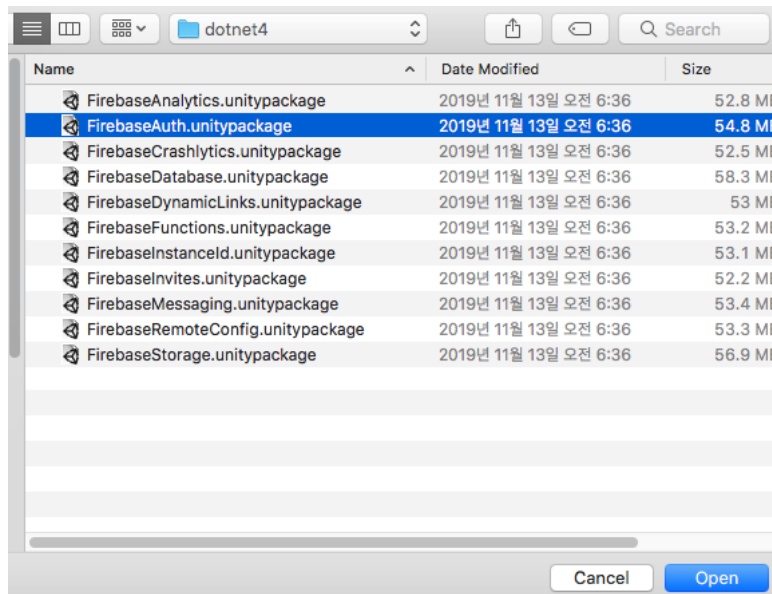
Client

아래 링크를 참고하여 Unity3d Client Build Setting 을 해준다.

https://seonbicode.tistory.com/43#Unity3d_Client_Setting

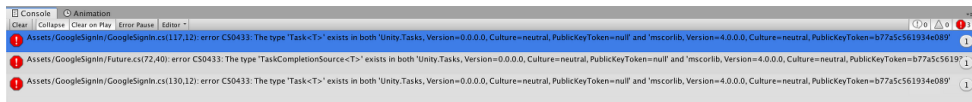
Assets 메뉴 - Import Package - Custom Package 로 다운로드 받은 firebase_unity_sdk 폴더로 간 후 firebase sdk 를 추가한다.

Firebase SDK 패키지 중 **FirebaseAuth.unitypackage** 를 추가한다.

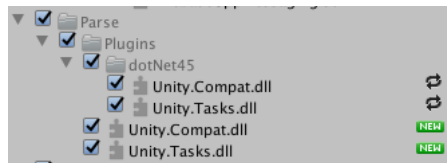


Player Settings 의 Scripting Runtime Version (.NET 버전) 을 확인하여,
 Net 3.5 면 dotnet3 폴더에 있는 FirebaseMessaging.unitypackage 파일을 임포트한다.
 Net 4.x 라면 dotnet4 폴더에 있는 FirebaseMessaging.unitypackage 파일을 임포트한다.

SDK Import 후 3개의 오류가 발생할 수 있다.



4.x 라면 dotnet45 폴더 바깥에 있는 Unity.Compat.dll 과 Unity.Tasks.dll 을 삭제해주고
 3.5 라면 dotnet45 폴더에 있는 Unity.Compat.dll 과 Unity.Tasks.dll 을 삭제해준다.



Unity3d에서 Google 로그인 인증을 사용하기 위해서는 [google-signin-unity](#) 이 포함되어야 한다.
 최신 버전의 unitypackage 를 다운로드 받은 후 Import 한다. (최신 버전 1.0.4)

Latest release

v1.0.4

588c063

Version 1.0.4

claywilkinson released this on 20 Sep 2018

Bug fixes for

- Adding additional scopes causes crash
- Handling non-success results in onActivityResult

Added arm64 binary library.

Assets 4

google-signin-plugin-1.0.4.unitypackage	533 KB
GoogleSignIn-sample.unitypackage	7.57 KB
Source code (zip)	
Source code (tar.gz)	

익명, 구글, 이메일/비밀번호 인증을 코드에서 처리하는 방법이다.

login.cs
 0.01MB

```

using Google;
using Firebase.Auth;

using UnityEngine;
using System.Threading.Tasks;

public class login : MonoBehaviour
{
    // Auth 용 instance
    FirebaseAuth auth = null;

    // 사용자 계정
    FirebaseUser user = null;

    // 기기 연동이 되어 있는 상태인지 체크한다.
    private bool signedIn = false;

    private void Awake()
    {
        // 초기화
        auth = Firebase.Auth.FirebaseAuth.DefaultInstance;

        // 유저의 로그인 정보에 어떠한 변경점이 생기면 실행되게 이벤트를 걸어준다.
        auth.StateChanged += AuthStateChanged;
        //AuthStateChanged(this, null);
    }

    // 계정 로그인에 어떠한 변경점이 발생시 진입.
    void AuthStateChanged(object sender, System.EventArgs eventArgs)
    {
        if (auth.CurrentUser != user)
        {
            // 연동된 계정과 기기의 계정이 같다면 true를 리턴한다.
            signedIn = user != auth.CurrentUser && auth.CurrentUser != null;

            if (!signedIn && user != null)
            {
                UnityEngine.Debug.Log("Signed out " + user.UserId);
            }

            user = auth.CurrentUser;

            if (signedIn)
            {
                UnityEngine.Debug.Log("Signed in " + user.UserId);
            }
        }
    }

    ////////////
    // 익명 로그인 //
    ////////////
    public void AnonyLogin()
    {
        // 익명 로그인 진행
        auth.SignInAnonymouslyAsync().ContinueWith(task =>
        {
            if (task.IsCanceled)
            {
                Debug.LogError("SignInAnonymouslyAsync was canceled.");
                return;
            }
            if (task.IsFaulted)
            {
                Debug.LogError("SignInAnonymouslyAsync encountered an error: " + task.Exception);
                return;
            }

            // 익명 로그인 연동 결과
            Firebase.Auth.FirebaseUser newUser = task.Result;
            Debug.LogFormat("User signed in successfully: {0} ({1})",
                newUser.DisplayName, newUser.UserId);
        });
    }

    ////////////
    // 구글 로그인 //
    ////////////

```

```

private void GoogleLoginProcessing()
{
    if (GoogleSignIn.Configuration == null)
    {
        // 설정
        GoogleSignIn.Configuration = new GoogleSignInConfiguration
        {
            RequestIdToken = true,
            RequestEmail = true,
            // Copy this value from the google-service.json file.
            // oauth_client with type == 3
            WebClientId = ""
        };
    }

    Task<GoogleSignInUser> signIn = GoogleSignIn.DefaultInstance.SignIn();

    TaskCompletionSource<FirebaseUser> signInCompleted = new TaskCompletionSource<Fire
    signIn.ContinueWith(task =>
    {
        if (task.IsCanceled)
        {
            Debug.Log("Google Login task.IsCanceled");
        }
        else if (task.IsFaulted)
        {
            Debug.Log("Google Login task.IsFaulted");
        }
        else
        {
            Credential credential = Firebase.Auth.GoogleAuthProvider.GetCredential(((T
            auth.SignInWithCredentialAsync(credential).ContinueWith(authTask =>
            {
                if (authTask.IsCanceled)
                {
                    signInCompleted.SetCanceled();
                    Debug.Log("Google Login authTask.IsCanceled");
                    return;
                }
                if (authTask.IsFaulted)
                {
                    signInCompleted.SetException(authTask.Exception);
                    Debug.Log("Google Login authTask.IsFaulted");
                    return;
                }

                user = authTask.Result;
                Debug.LogFormat("Google User signed in successfully: {0} ({1})", user.DisplayName
                return;
            }));
        }
    });
}

//////////
// 이메일 로그인 //
//////////
public void EmailLogin()
{
    // 적당한 UGUI 를 만들어 email, pw 를 입력받는다.
    var email = EmailCreatePanel.transform.Find("email").Find("Text").GetComponent<UnityF
    var pw = EmailCreatePanel.transform.Find("pw").Find("Text").GetComponent<UnityEngin

    if (email.Length < 1 || pw.Length < 1)
    {
        Debug.Log("이메일 ID 나 PW 가 비어있습니다.");
        return;
    }

    auth.CreateUserWithEmailAndPasswordAsync(email, pw).ContinueWith(task =>
    {
        if (task.IsCanceled)
        {
            UnityEngine.Debug.LogError("CreateUserWithEmailAndPasswordAsync was canceled.")
            return;
        }
        if (task.IsFaulted)
        {
            UnityEngine.Debug.LogError("CreateUserWithEmailAndPasswordAsync encountered ar

```

```

        return;
    }

    // firebase email user create
    Firebase.Auth.FirebaseUser newUser = task.Result;
    UnityEngine.Debug.LogFormat("Firebase Email user created successfully. {0} ({1})", newUser .
    return;
    });
}

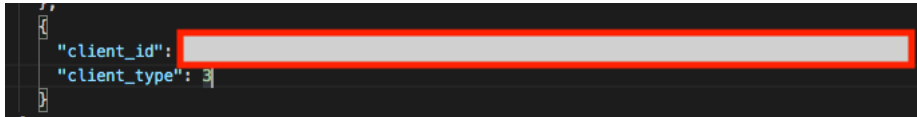
// 연동 해제
public void SignOut()
{
    if (auth.CurrentUser != null)
        auth.SignOut();
}

// 연동 계정 삭제
public void UserDelete()
{
    if (auth.CurrentUser != null)
        auth.CurrentUser.DeleteAsync();
}
}

```

이메일/비밀번호 로그인 방식은 유저가 이메일과 비밀번호를 입력할 수 있는 UI 를 만들어준 후 이메일과 비밀번호를 입력받아야 한다.

구글로그인에는 webClientID 가 필요하다. 이 아이디는 google-services.json 에서 구할 수 있다.



UGUI 버튼이나 어느 특정 트리거를 이용하여 각 로그인 함수를 불러 사용할 수 있다.

추가로 익명 로그인 상태인 유저를 구글 로그인 or 이메일 로그인 으로 변화시키는 방법이다.

AnonyOfChange.cs

AnonyOfChange.cs
0.00MB

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

using Google;
using Firebase;
using Firebase.Auth;
using Firebase.Unity.Editor;
using UnityEngine.UI;
using System.Threading.Tasks;

public class AnonyOfChange : MonoBehaviour
{
    // Auth 용 instance
    FirebaseAuth auth = null;

    // 사용자 계정
    FirebaseUser user = null;

    // 이메일 전환창
    public GameObject emailPanel;

    private void Start()
    {
        auth = Firebase.Auth.FirebaseAuth.DefaultInstance;
    }
}

```

```

    }

    // 익명 로그인 -> 구글 로그인
    public void onAnonyToGoogle()
    {
        if (auth.CurrentUser != null)
        {
            Debug.Log(auth.CurrentUser.UserId);

            if (GoogleSignIn.Configuration == null)
            {
                GoogleSignIn.Configuration = new GoogleSignInConfiguration
                {
                    RequestIdToken = true,
                    RequestEmail = true,
                    // Copy this value from the google-service.json file.
                    // oauth_client with type == 3
                    WebClientId = ""
                };
            }

            Task<GoogleSignInUser> signIn = GoogleSignIn.DefaultInstance.SignIn();
            TaskCompletionSource<FirebaseUser> signInCompleted = new TaskCompletionSource<FirebaseUser>

            signIn.ContinueWith(task =>
            {
                if (task.IsCanceled)
                {
                    Debug.Log("Google Login task.IsCanceled");
                }
                else if (task.IsFaulted)
                {
                    Debug.Log("Google Login task.IsFaulted");
                }
                else
                {
                    Credential credential = Firebase.Auth.GoogleAuthProvider.GetCredential(((Task<
                    //string currentUserId = auth.CurrentUser.UserId;
                    //string cureentEmail = auth.CurrentUser.Email;
                    //string currentDisplayName = auth.CurrentUser.DisplayName;
                    //System.Uri currentPhotoUrl = auth.CurrentUser.PhotoUrl;

                    auth.CurrentUser.LinkWithCredentialAsync(credential).ContinueWith(authTask =>
                    {
                        if (authTask.IsCanceled)
                        {
                            signInCompleted.SetCanceled();
                            Debug.Log("Google Login authTask.IsCanceled");
                            return;
                        }
                        if (authTask.IsFaulted)
                        {
                            signInCompleted.SetException(authTask.Exception);
                            Debug.Log("Google Login authTask.IsFaulted");
                            return;
                        }

                        user = authTask.Result;
                        Debug.LogFormat("Google User signed in successfully: {0} {1}", user.DisplayName, use
                    });
                }
            });
        }
        else
        {
            Debug.Log("Not logged in");
        }
    }

    // 익명 로그인 -> 이메일 로그인
    public void onAnonyToEmail()
    {
        if (emailPanel == null)
        {
            Debug.Log("이메일 전환창이 없습니다.");
            return;
        }

        // 아래에 위치한 onEmailChangeSwich() 를 실행시키기 위한 패널
        emailPanel.SetActive(true);
    }

    // 이메일 로그인 정보를 받아온 후 사용한다.
    public InputField id;
    public InputField pw;
    public void onEmailChangeSwich()
    {
        if (id.text.Length < 1 || pw.text.Length < 1)
        {
            Debug.Log("이메일 ID 나 PW 가 비어있습니다.");
            return;
        }
    }

```

```

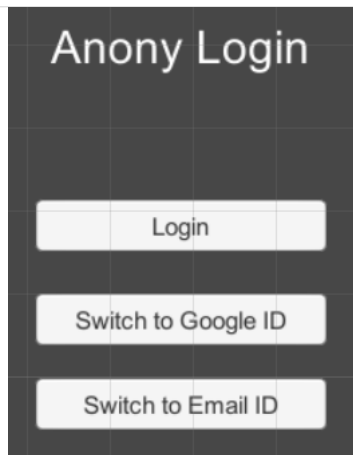
    }

    Credential credential = Firebase.Auth.EmailAuthProvider.GetCredential(id.text, pw.text);

    auth.CurrentUser.LinkWithCredentialAsync(credential).ContinueWith(task =>
    {
        if (task.IsCanceled)
        {
            Debug.Log("Email Login task.IsCanceled");
            return;
        }
        if (task.IsFaulted)
        {
            Debug.Log("Email Login task.Faulted");
            return;
        }

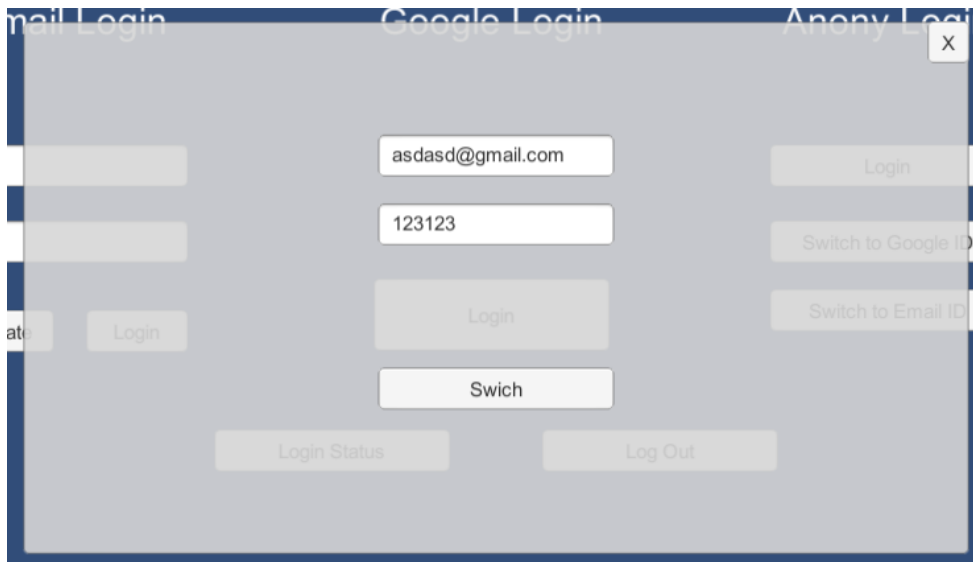
        user = task.Result;
        Debug.LogFormat("Firebase Email user created successfully: {0} <{1}>", user.DisplayName, user.UserId);
    });
}
}

```



Switch to Google ID 를 누르면 onAnonyToGoogle() 함수가 실행되며 구글 로그인 창이 활성화 되며 로그인 정보를 변경할 구글 계정을 누르면 익명 로그인이 구글 로그인으로 변경된다.

Switch to Email ID 를 누르면 onAnonyToEmail() 함수가 실행되며 아래와 같은 이메일, 비밀번호 정보를 받는 창이 활성화 된다.



위와 같은 창에서 이메일과 비밀번호를 입력후 Switch 버튼을 누르면 onEmailChangeSwich() 함수가 실행되며

(익명)		2019. 12. 24.	2019. 12. 24.	w2KefLIA48aUuDqN4mDUNatC1...
------	--	---------------	---------------	------------------------------

이랬던 익명 로그인 정보가...

asdasd@gmail.com		2019. 12. 24.	2019. 12. 24.	w2KefLIA48aUuDqN4mDUNatC1...
------------------	--	---------------	---------------	------------------------------

이렇게 이메일 로그인으로 변경되지만 UID는 그대로 보전된다.

여기서 핵심인 부분은 아래와 같다.

```
// 익명 -> 구글
Credential credential = Firebase.Auth.GoogleAuthProvider.GetCredential(((Task<GoogleSignIn
auth.CurrentUser.LinkWithCredentialAsync(credential).ContinueWith(authTask => {}));

// 익명 -> 이메일
Credential credential = Firebase.Auth.EmailAuthProvider.GetCredential(id.text, pw.text);
auth.CurrentUser.LinkWithCredentialAsync(credential).ContinueWith(task => {});
```

공감


구독하기


'Programming > Firebase' 카테고리의 다른 글


[Firebase] 서버에서 Database 사용하기	2019.12.17
[Firebase] Unity3d 인앱 결제 + 서버에 영수증 저장	2019.12.17
[Firebase] Unity3d 유저 인증	2019.12.17
[Firebase] Firebase 기본 설정	2019.12.17
[Firebase] Push (Cloud Messaging) 사용하기	2019.12.13
[Firebase] Firebase Hosting 기본 제작	2019.12.05




Programming/Firebase 관련 글

- 

[Firebase] 서버에서 Database 사용하기
2019.12.17
- 

[Firebase] Unity3d 인앱 결제 + 서버에 영수증 저장
2019.12.17
- 

[Firebase] Firebase 기본 설정
2019.12.17
- 

[Firebase] Push (Cloud Messaging) 사용하기
2019.12.13

글 더보기

0

☐ 비공개

작성자명

비밀번호

소중한 댓글 하나가 블로거에게 힘이 됩니다.

댓글 등록

«	1	...	11	12	13	14	15	16	17	18	19	...	48	»
---	---	-----	----	----	----	----	----	----	----	----	----	-----	----	---

🔍 검색어 입력 후 엔터를 누르세요.

❗ 공지사항

➤ 공지사항

≡ 전체 카테고리

Programming

➤ C&C++
➤ Unity3D
➤ C#
➤ Firebase
➤ Server
➤ Tool
➤ Docker

마이취미

➤ 내사내리
➤ 가지고놀이

제작

➤ 만드느중
➤ 만든거

JustDolt

잡다한 자료(번역)
➤ WinForm+Unity3D
➤ Unity3D
➤ C&C++
➤ C#

애드센스 광고 영역

최근 글

최근 댓글

Unity Package Manager 로 관리하기
Blazor Server에서 npm 패키지 사용하기
서피스북2에 window10, ubuntu20.04 ...
Containerising Blazor Applications Wi...
xcodeproj에서 구문에러가 발생할때
라즈베리파이예 Gitlab 구축하기
[Storage] Firebase Storage를 CDN 처럼...
Addressable 관련 정리
Unity3d 오답노트
AssetBundle 관련 정리

🏷 태그











#firebase storage #Firebase

#Firebase Database #Unity3D #tool

#web

더보기+

🔍 전체 방문자	
오늘	138
어제	144
전체	91,721

🔖 블로그 인기글	
	Unity + WinForm 을 구현해보자! 1편
	How To Draw Line On Screen In Unity (C#)
	문자열 자르기
	[Firebase] Unity3d에서 로그인 + 인앱...
	C++ 의 포인터 데이터를 C#으로 전달하기
	[Firebase] Unity3d 인앱 결제 + 서버에...
	FILE MAPPING : Memory Mapped File (MMF)
	[Firebase] Web에서 Database 사 용하기
	Addressable 관련 정리
	Asynchronous Client/Server Socket Exa...