

Chapter. 01

**Firebase**를 이용한 클라이언트 네트워크 구현하기

# | Firebase 활용하기

FAST CAMPUS  
ONLINE

Firebase 활용하기

강사. 김영민

Firebase 소개

Firebase 설정하기

사용자 인증 구현하기

사용자 순위 (leaderboard) 구현하기

사용자 데이터 저장/불러오기

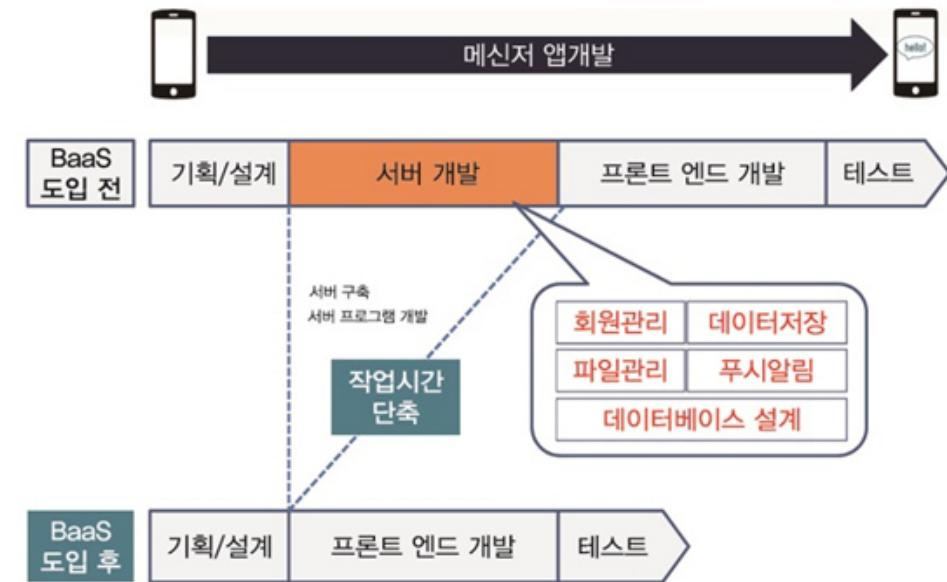
# I Firebase 소개

## Firebase는?

- Backend as a Service (BaaS)
- Android, iOS, Web, Unity 지원
  - (일부 기능은 특정 플랫폼 지원)
- 기반 데이터 포맷: JSON

## Firebase 특징

- 개발자가 개발 속도를 높이면서 사용자에게 집중
- Google 인프라 위에서 자동확장
- 데이터와 통계를 서로 공유



# I Firebase 소개

## Firebase에서 지원하는 백엔드 서비스들

- Authentication
- Database: Realtime Database or Cloud Firestore
- Cloud Functions
- Hosting

## Firebase 운영 지원 서비스들

- Google Analytics
- Crashlytics
- Could Messaging
- Remote Config

# I Firebase 설정하기

1. Firebase에 가입하기: <https://firebase.google.com>
2. 프로젝트 추가하기: <https://console.firebaseio.google.com>
3. 앱 추가하기: Unity -> Android
4. 구성 파일 다운로드
5. Firebase SDK 다운로드

# I 사용자 인증 구현하기

## 1. 인증 설정하기

The screenshot shows the Firebase Authentication console for a project named 'FastCampus'. On the left sidebar, the 'Authentication' icon is highlighted with a red box and labeled '선택' (Selected) in red text. The main tab 'Sign-in method' is also highlighted with a red box. The 'Email/Password' sign-in method is listed at the top of the 'Sign-in methods' section, with its status '사용 설정됨' (Enabled) also highlighted with a red box.

제공업체	상태
이메일/비밀번호	사용 설정됨
전화	중지됨
Google	중지됨
Play 게임	중지됨
게임 센터 <small>Beta</small>	중지됨
Facebook	중지됨
Twitter	중지됨
Github	중지됨
Yahoo	중지됨
Microsoft	중지됨
Apple	중지됨
익명	중지됨

# | 사용자 인증 구현하기

## 2. Unity SDK 설치하기

1. FirebaseAnalytics.unitypackage 설치
2. FirebaseAuth.unitypackage 설치

# | 사용자 인증 구현하기

## 1. Firebase.Auth.FirebaseAuth 클래스 접근

```
Firebase.Auth.FirebaseAuth auth = Firebase.Auth.FirebaseAuth.DefaultInstance;
```

## 2. 계정 생성하기

```
auth.CreateUserWithEmailAndPasswordAsync(email, password).ContinueWith(task => {  
});
```

## 3. 로그인 처리하기

```
auth.SignInWithEmailAndPasswordAsync(email, password).ContinueWith(task => {  
});
```

## 4. 프로필 정보 접근하기

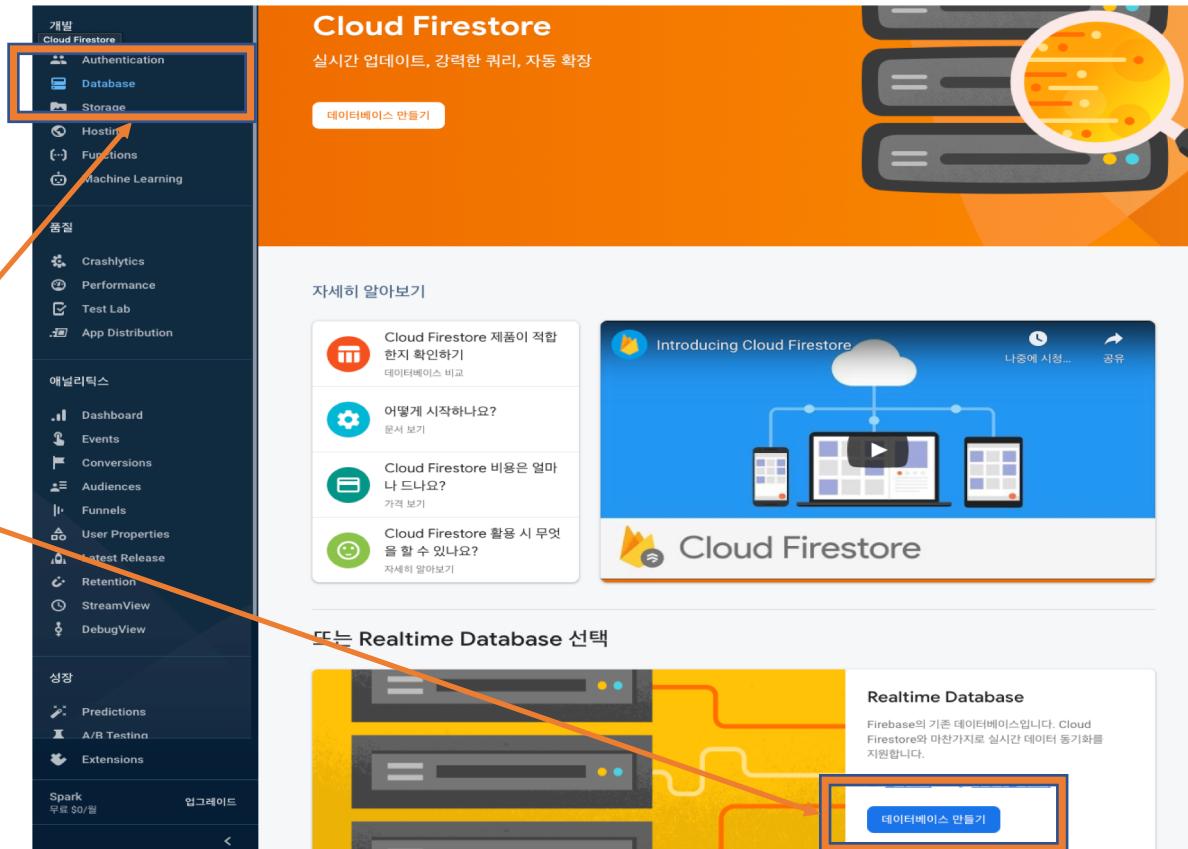
```
Firebase.Auth.FirebaseUser user = auth.CurrentUser;
```

```
FirebaseUser.DisplayName, Email, PhotoUrl, UserId
```

# I 사용자 순위 (leaderboard) 구현하기

## 1. 데이터 베이스 설정하기

선택



## 2. Unity SDK 설치하기

### 1. FirebaseDatabase.unitypackage 설치

# I 사용자 순위 (leaderboard) 구현하기

## 1. 유니티 데이터 베이스 접근 구성

- `FirebaseApp.DefaultInstance.SetEditorDatabaseUrl("https://YOUR-FIREBASE-APP.firebaseio.com/");`
- `FirebaseApp.DefaultInstance.Options.DatabaseUrl;`

## 2. DatabaseReference

```
DatabaseReference reference = FirebaseDatabase.DefaultInstance.RootReference;
```

## 3. 데이터 쓰기

```
mDatabaseRef.Child("users").Child(userId).Child("username").SetValueAsync(name);
```

## 4. 데이터 업데이트

```
mDatabase.UpdateChildrenAsync(childUpdates);
```

## 5. 데이터 삭제

```
RemoveValue() or SetValueAsync(null) or UpdateChidrenAsync(null)
```

# I 사용자 순위 (leaderboard) 구현하기

## 1. 데이터 읽기

```
FirebaseDatabase.DefaultInstance.GetReference("Leaders").GetValueAsync().ContinueWith(task => {  
});
```

## 2. 데이터 변경 이벤트 등록

```
FirebaseDatabase.DefaultInstance.GetReference("Leaders").ValueChanged += HandleValueChanged;
```

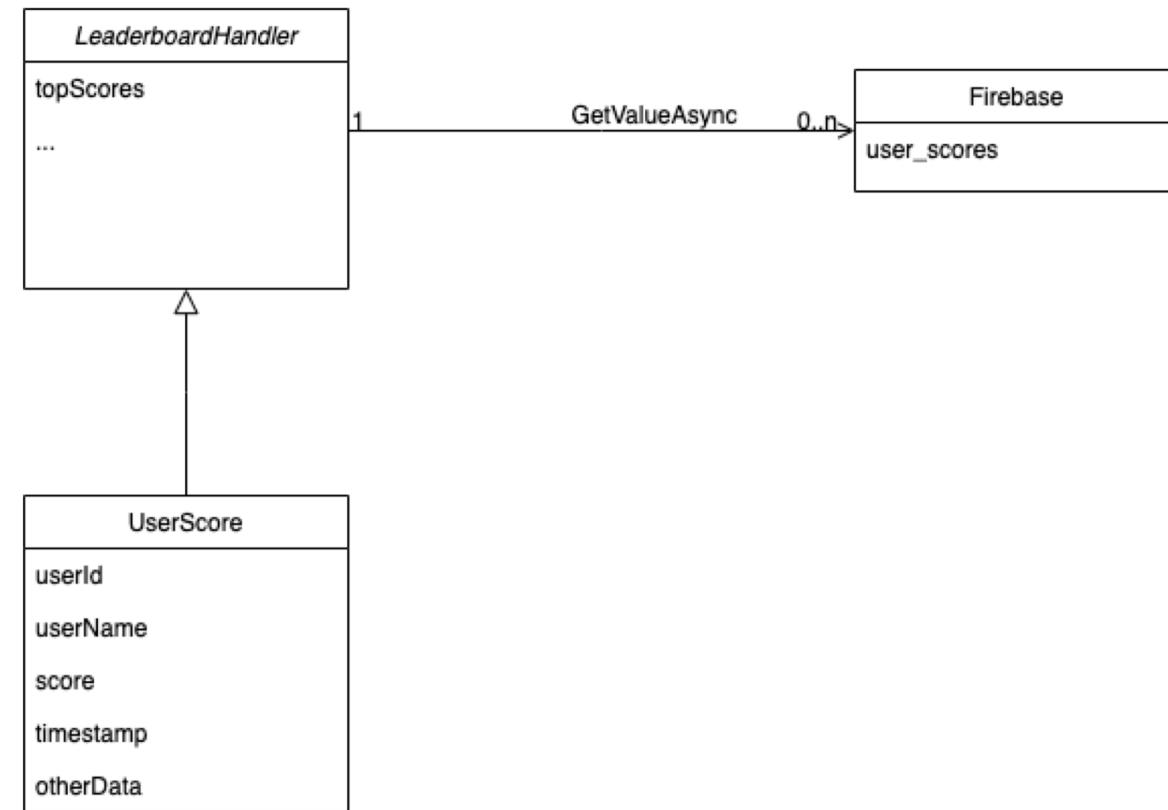
## 3. 데이터 정렬: OrderByChild(), OrderByKey(), OrderByValue()

```
FirebaseDatabase.DefaultInstance.GetReference("Leaders").OrderByChild("score")
```

## 4. 데이터 필터링: StartAt(), EndAt(), EqualTo()...

```
FirebaseDatabase.DefaultInstance.GetReference("Leaders").OrderByChild("score").StartAt(1)
```

# I 사용자 순위 (leaderboard) 구현하기



# I 사용자 데이터 저장/불러오기 - 보안 규칙

## 규칙 유형

.read	데이터를 읽을 수 있는 조건
.write	데이터를 쓸 수 있는 조건

```
{
  /* Visit https://firebase.google.com/docs/database/security to learn more about security rules. */
  "rules": {
    "all_scores": {
      ".read": "true",
      ".write": "true",
    },
    "users": {
      "$uid": {
        ".read": "auth != null",
        ".write": "auth != null && $uid === auth.uid",
      }
    }
  }
}
```

# I 사용자 데이터 저장/불러오기 - 데이터 검증과 색인 정의

## 규칙 유형

.validate	값의 올바른 형식, 하위 속성을 갖는지 여부 및 데이터 유형 정의
.indexOn	정렬 및 쿼리를 위해 색인화할 하위 항목 지정

```
{
  /* Visit https://firebase.google.com/docs/database/security to learn more about security rules. */
  "rules": {
    "all_scores": {
      ".read": "true",
      ".write": "true",
      // data written to /all_score must be a number greater than 0
      ".validate": "newData.isNumber() && newData.val() > 0",
      // index by score
      ".indexOn": ["score"]
    },
    "users": {
      "$uid": {
        ".read": "auth != null",
        ".write": "auth != null && $uid === auth.uid",
      }
    }
  }
}
```

# | 사용자 데이터 저장/불러오기 - Newtonsoft Json 소개

## 1. JsonUtility

- Serialize/Desrialize 에 대한 최소한의 기능 제공
- Vector serialize
- MonoBehaviour Serialize: FromJsonOverwrite()
- Dictionary 지원

## 2. Newtonsoft JSON

- <https://github.com/JamesNK/Newtonsoft.Json>
- <https://github.com/JamesNK/Newtonsoft.Json/releases>
- 최신 Release.zip를 다운로드
- net 45 폴더의 NetwonsoftJson.dll -> Assets 폴더로 복사
- 세세한 설정 가능
- BSON 지원
- Properties와 Field에 세세한 설정 필요
- MonoBehaviour와 Vector 형등 기본 지원 클래스에 대한 설정 필요

# | 사용자 데이터 저장/불러오기

- JSON을 사용한 데이터 쓰기

```
FirebaseDatabase.DefaultInstance.Child(...).SetRawJsonValueAsync(statsJson).ContinueWith(task =>
{
    ...
});
```

- JSON을 사용한 데이터 읽기

```
FirebaseDatabase.DefaultInstance.Child(...).GetValueAsync().ContinueWith(task =>
{
    ...
    DataSnapshot snapshot = task.Result;
    snapshot.GetRawJsonValue();
});
```