
웹&DB서버 설계 및 구동

N A M E : 권 성 호

T E L : 010-8874-6452

P A R T : PROGRAMING

Efforts make all doors open



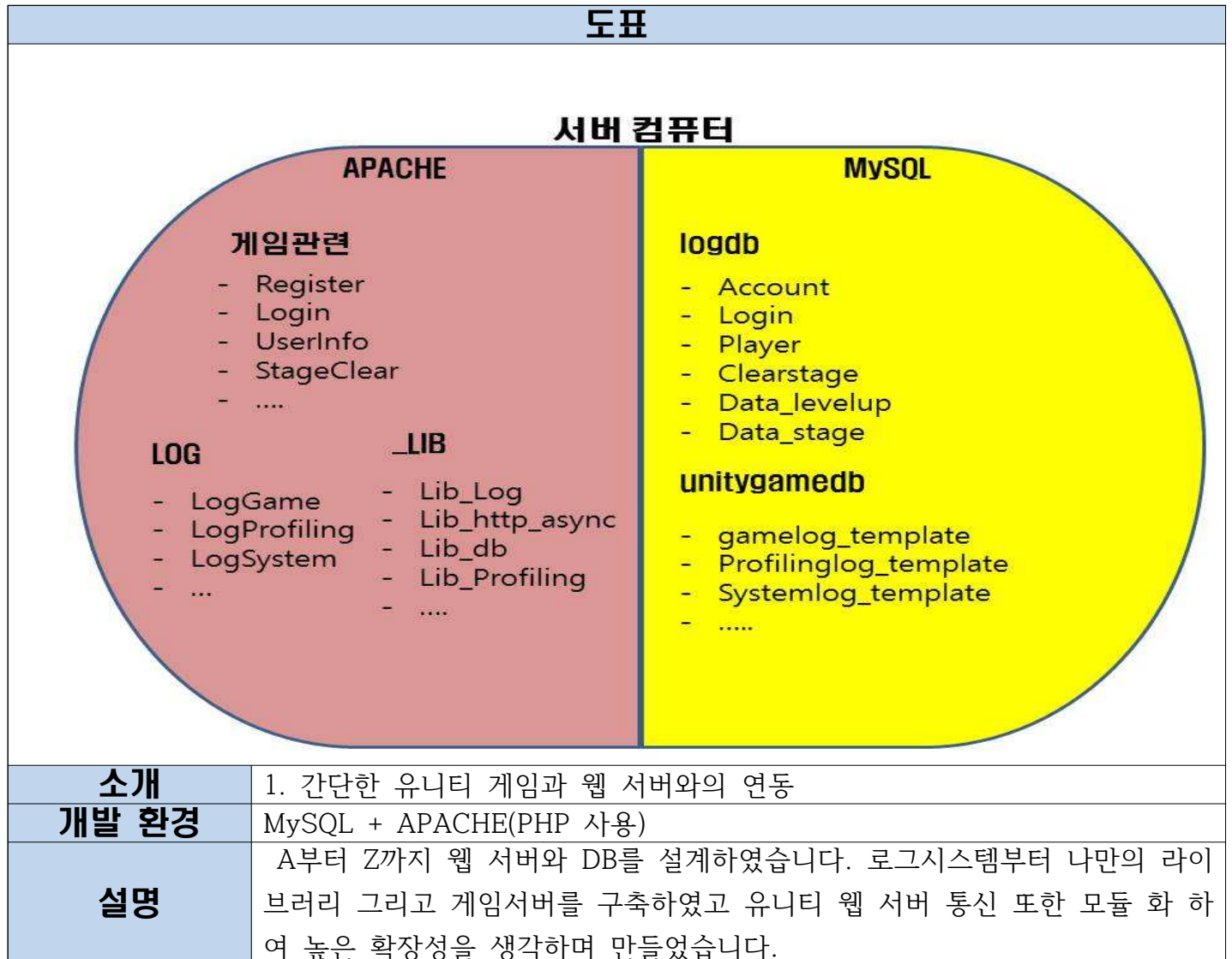
목 차



I . 개요	1
1. 설계 도표	1
2. DB 구성.....	2
3. 웹 서버(PHP문 구성).....	4
4. 유니티 웹 모듈.....	7
5. 유니티 클라이언트	11
6. 다운로드 링크	12

I. 개요

1. 설계 도표



2. DB 구성

DB	logdb
테이블 구성	
gamelog_ template	게임의 로직의 로그를 저장합니다. - 컬럼 no - 기본키 date - 시간 AccountNo - 사용자 번호 LogType - 큰 게임단락 LogCode - 작은 소목록 Param1~4 - 게임로직 인덱스(기획자와 상담) ParamString - 기타 저장
systemlog_ template	게임시스템의 로그를 저장합니다. no - 기본키 date - 시간 AccountNo - 사용자 번호 Action - 호출 함수 Message - 호출 문
profilinglog_ template	게임 서버 PHP문의 모듈별 시간을 측정합니다.(100의 1번꼴) no - 기본키 date - 시간 IP - 아이피 AccountNo - 사용자번호 Action - 호출 서버(페이지 이름) T_Page - 전체시간 T_Mysql_Conn - DB연결시간 T_Mysql - 쿼리 응답시간 T_ExtAPI - 외부 API모듈시간 T_Log - 로그 자체 시간 Query - 쿼리 시간 측정시 쿼리문 Comment - 전체적인 스트링(무음)
설명	
템플릿 테이블을 설계하여 매 달마다 테이블을 템플릿 테이블과 같은 컬럼으로 웹 서버에서 생성합니다. 예)systemlog_201609 게임 제작 및 라이브에 있어 로그저장은 매우 중요한 과정입니다. 게임 제작중 예외상황을 한 눈에 파악할 수 있으며 라이브 중 사용자의 이상행동 및 행동패턴을 파악할 수 있습니다.	

DB	unitygamedb
테이블 구성	
account	사용자 계정을 저장 accountno - 사용자 고유 번호 id - 사용자 id password - hash로 변환된 패스워드
clearstage	유저 게임 클리어 정보를 저장 accountno - 사용자 번호 stageid - 스테이지 키
data_levelup	레벨업에 필요한 경험치 저장 level - 레벨 exp - 다음 레벨을 위한 경험치
data_stage	스테이지 정보 저장(보상) stageid - 스테이지 키 clearexp - 스테이지 경험치
login	로그인 정보 저장 accountno - 사용자 번호 time - timestamp ip - 사용자 ip count - 로그인 횟 수(전체)
player	유저 정보 accountno - 사용자 번호 level - 사용자 레벨 exp - 현재 경험치
session	섹션 키를 저장합니다.(세션을 통해 로그인 상태인지 확인합니다. 일정 시간 세션 키를 발급 후 5분이 지나면 발급 받은 섹션 키를 사용할 수 없습니다.) accountno - 사용자 번호 session - 사용자 고유 hash값 time - timestamp(5분 체크 용)
system	버전 확인 용(클라이언트의 버전을 확인하기 위한 테이블로 버전이 다르면 게임의 로직을 처리하지 않습니다) versionmajor - 메이저번호 versionminor - 마이너번호
설명	
웹 서버에서 실제 필요한 DB입니다. 웹 서버는 클라이언트의 상태를 저장 할 수 없습니다. 그러므로 세션 키를 통해 사용자를 확인하고 게임의 로직을 처리합니다.	

3. 웹 서버(PHP문 구성)

폴더 명	_LIB
PHP파일	
_Config_DB	- DB연결에 필요 한 변수들을 저장
_Config_Log	- Log저장에 필요 한 URL주소 와 프로파일링 로그 확률 로그 Level을 지정
lib_Common	ResponseJSON - 배열을 JSON형식으로 만드는 함수 QuitError - 에러코드를 JSON형식으로 만드는 함수
lib_db	DB_TransactionQuery - Query를 배열형식으로 모아 트랜잭션처리를 하여 보냄 - 실패 시 롤백 DB_ExecQuery - 일반 쿼리를 보내는 모듈
lib_ErrorHandler	error_reporting(E_ALL) set_error_handler('ERROR_Handler') set_exception_handler('EXCEPTION_Handler') register_shutdown_function('sys_Shutdown') - Error발생 시 로그에 저장
lib_http_async	curl_request_async - HTTP 헤더를 만들어 보냄 - 응답이 필요 없는 패킷을 보낼 때 사용 - POST,GET방식 지정 가능
lib_Log	Log_System - Log PHP문을 실행 CGameLog - 싱글톤으로 만들어진 GameLog클래스
lib_Profiling	Profiling - 프로파일링 싱글톤 클래스
설명	
웹 서버의 전체적인 모듈로 이루어진 폴더 필요 시 include하여 사용	

폴더 명	LOG
PHP파일	
_Config	- logdb의 연결 시 필요 변수 저장
_ResultCode	- 각 오류코드마다의 오류String(적용 하지는 못했음//시간 부족)
LogGame	- JSON 형식으로 들어 온 게임 정보를 logdb에 저장
LogProfiling	- JSON 형식으로 들어 온 프로파일링 정보를 logdb에 저장
LogSystem	- JSON 형식으로 들어 온 시스템 정보를 logdb에 저장
설명	
로그를 저장하기 위한 하나의 웹 서버라 생각하면 됨(장비 부족으로 폴더로만 구분)	

폴더 명	public_html
PHP파일	
_Config_DB	- unitygamedb의 연결 시 필요 변수 저장
_Startup	- 모듈 Include - JSON 형식 확인 - AccountNo, Version 확인
Register	Request { "id" : "사용아이디", "password" : "사용패스워드" } Response { ..공통.. }
Login	Request { "id" : "사용아이디", "password" : "사용패스워드" } Response { ..공통.. "accountno" : no "session" : "세션키" }
StageClear	Request { "accountno" : bigint, "session" : "세션", "stageid" : stageid, } Response { ..공통.. "exp" : 경험치 (내 보유 경험치) "level" : 레벨 (내 레벨) } * data_stage 에 stageid 의 존재여부 확인 * 유저의 클리어 여부 확인 후 경험치 지급 * 레벨업 처리 까지.
UserInfo	Request { "accountno" : bigint, "session" : "세션", } Response { ..공통.. "exp" : 경험치 "level" : 레벨 }
_Cleanup	-DB클로즈 -로그서버로 게임 로그 전송
설명	
실제 게임 웹 서버	

* 모든 Response 의 기본형식

```
{
    "ResultCode" : 결과코드 int,
    "ResultMsg"  : "결과 메시지"

    ... 기타 메시지...
}
```


4. 유니티 웹 모듈

Globals.cs	<pre> 1 using UnityEngine; 2 using System.Collections; 3 4 5 namespace Globals 6 { 7 public static class INSTANCE 8 { 9 public static string sessionkey; 10 public static int accountno = 0; 11 public static int nowstage = 1; 12 13 public static int exp; 14 public static int level; 15 16 public const string version = "1\r\n1\r\n"; 17 } 18 } </pre>
코드분석	<p>웹 서버에서 받은 내용 및 게임 전반적인 내용을 저장하기 위한 글로벌 클래스 웹 서버에서 받은 sessionkey, accountno, exp, level을 저장한다. version에 개행 문자 를 넣은 이유는 POST형식으로 웹 서버와 통신하며 majorVersion\r\nminorVersion\r\naccountno\r\nJSON형식의 패킷을 주고 받는 형식 으로 서버에서는 \r\n을 구분으로 버전과 사용자 번호 JSON스트링을 구분한다.</p>

DataClass.cs

```

1  using UnityEngine;
2  using System;
3  using System.Collections;
4  using System.Collections.Generic;
5  using System.Text;
6  using LitJson;
7  using UnityEngine.UI;
8
9  public interface IWebData
10 {
11     void Recv(JsonData jsonObject);
12     string URL();
13 }
14
15 public class WebAccountRegister...
16
17 public class WebUserInfo...
18
19 public class WebLogin:IWebData
20 {
21     public string id { get; set; }
22     public string password { get; set; }
23
24     public string URL() { return "Login.php"; }
25
26     public void Recv(JsonData jsonObject)
27     {
28         //받은 세션키 저장
29         Globals.INSTANCE.sessionkey = jsonObject["session"].ToString();
30         Globals.INSTANCE.accountno = Convert.ToInt32(jsonObject["accountno"].ToString());
31
32         Debug.Log("Login Success");
33
34         Globals.INSTANCE.nowstage = 1;
35         Application.LoadLevel("StageA");
36     }
37 }
38
39 public class WebStageClear...

```

코드분석

웹통신 NetHTTP클래스를 사용하는 데이터 클래스.(패킷 클래스)
 하나의 PHP파일마다, 요청마다 각각의 데이터 클래스를 하나씩 만드는게 편리하다. 각각의 클래스에서 자신의 결과에 대한 JSON파싱은 알아서 하여야 한다.

NetHTTP.cs (멤버 변수)

```

8  public class NetHTTP : MonoBehaviour {
9
10     const string PHP_URL = "http://procademyserver.ip1time.org:10401/";
11
12     static List<IWebData> m_SendList = new List<IWebData>();
13
14     public bool _BlockInput = false;
15     bool _MsgBox = false;
16     string _MsgString;
17
18     protected static NetHTTP _instance;

```

코드분석

기본 PHP_URL(현재 다니고 있는 학원에서 제공하는 개인 서버의 도메인/포트포워딩 되어 있다)과 데이터 클래스를 관리하는 리스트, 응답이 오기까지 키 입력을 막는 플래그 변수와 완료/실패 메시지를 위한 변수 등이 있고 싱글 톤 클래스를 위한 static변수가 있다.

NetHTTP.cs (SendURL)

```
IEnumerator SendURL(IWebData SendData)
{
    Debug.Log((PHP_URL + SendData.URL()));

    UTF8Encoding ue = new UTF8Encoding();
    string JSONString = JsonMapper.ToJson(SendData);
    Debug.Log(Globals.INSTANCE.version.ToString() +
        Globals.INSTANCE.accountno.ToString() + "\r\n" + JSONString);
    byte[] bytes = ue.GetBytes
        (Globals.INSTANCE.version.ToString() + Globals.INSTANCE.accountno.ToString()
        + "\r\n" + JSONString);
    Debug.Log(PHP_URL + SendData.URL());
    WWW www = new WWW(PHP_URL + SendData.URL(), bytes);
    yield return www;

    Response(SendData, (www.error != null));
    if(www.error == null)
    {

        Debug.Log(www.text);
        JsonData JsonResponse = JsonMapper.ToObject(www.text);

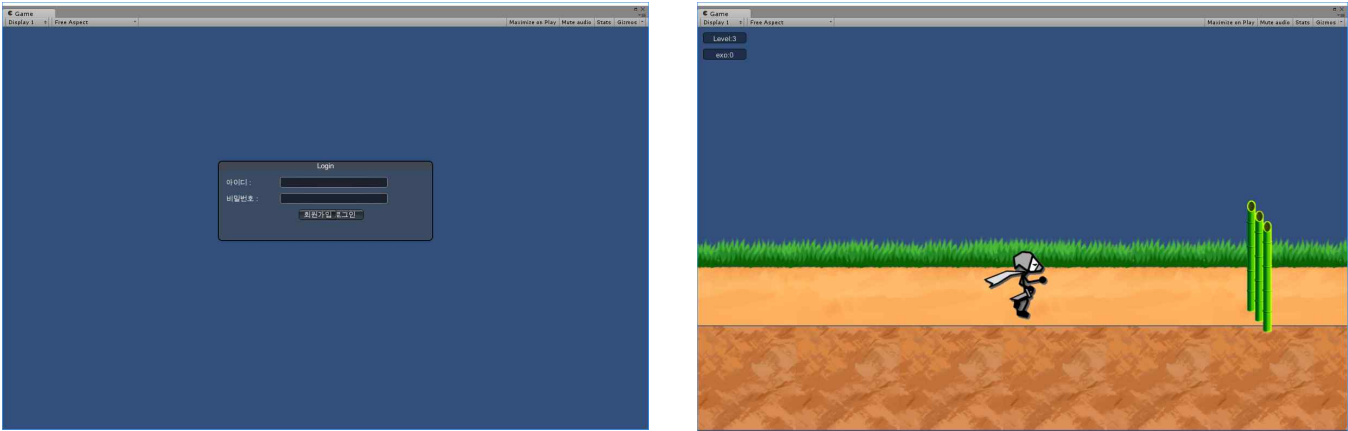
        int ResultCode = Convert.ToInt32(JsonResponse["ResultCode"].ToString());
        string ResultMsg = JsonResponse["ResultMsg"].ToString();

        if(ResultCode == 1)
        {
            SendData.Recv(JsonResponse);
        }
        else
        {
            MessageBox(ResultMsg);
            Debug.Log("url:" + SendData.URL() +
                "\nResultCode:" + ResultCode + "/ResultMsg:" + ResultMsg);
        }
    }
    else
    {
        Debug.Log("url:" + SendData.URL() + "\nerror:" + www.error);
    }
}
}
```

코드분석

실질적으로 코루틴 을 돌면 서 웹 서버와 통신하는 모듈이다. 보낼 DataClass를 JSON으로 변환 한 후 버전과 사용자 번호 그리고 JSON으로 변환 된 DataClass를 웹 서버에 송신한다. www객체는 응답이 올 때 까지 리턴하다 응답이 오면 응답이 온 JSONString을 DataClass에 보낸다.

5. 유니티 클라이언트

프로그램 명	WebRunningGame
플랫폼	PC
프로그램 화면	
	
소개	<ol style="list-style-type: none"> 1. 웹 서버 테스트를 위한 간단한 런닝게임 2. 회원가입 후 로그인을 하면 게임이 실행된다. 3. Stage가 끝날 때마다 결과가 서버에 반영된다.
개발 환경	Unity5.3
설명	<p>디버깅용 Test런닝게임</p> <p>현재 Player테이블에 공백 유저가 있어 로그인을 하면 게임이 실행된다. 회원가입 후 로그인을 하면 개인 계정이 생긴다. 비밀번호는 sha256으로 저장되므로 보안에는 문제가 없다.</p>

6. 다운로드 링크

php문서 :

<https://drive.google.com/file/d/0BzusZZDBAoL3bVFKbjNxSGxyNIU/view?usp=sharing>

런닝게임 실행파일:

<https://drive.google.com/file/d/0BzusZZDBAoL3a2hjN1VfZ3M4OFk/view?usp=sharing>

런닝게임 프로젝트 파일:

<https://drive.google.com/file/d/0BzusZZDBAoL3MnRQc21DeEh5N1k/view?usp=sharing>