1. 附件

* EzWebGame Model

class AuthModel extends CI\_Model

{

/\*\*

\* AuthModel::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->database();

}

/\*\*

\* AuthModel::getNextCommuKey()

\*

\* 確認通訊 Key 並產生下一組通訊 Key 並儲存起來

\*

\* @param mixed $cKey

\* @param mixed $out

\* @return

\*/

public function getNextCommuKey($cKey, $out)

{

if ($this->checkCommuKey($cKey))

{

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$nextCKey = $this->commuKeygen($userId, $gameId, $roomId);

$this->saveCommuKey($userId, $gameId, $nextCKey);

$out->save("cKey", $nextCKey);

return $nextCKey;

}

$out->wrong("Communication Key Deny");

}

/\*\*

\* AuthModel::editCommuKey()

\*

\* 修改溝通Key中所帶房間資訊

\*

\* @param mixed $cKey

\* @param mixed $iRoomId

\* @param mixed $out

\* @return

\*/

public function editCommuKey($cKey, $iRoomId, $out)

{

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$nextCKey = $this->commuKeygen($userId, $gameId, $iRoomId);

$this->saveCommuKey($userId, $gameId, $nextCKey);

$out->save("cKey", $nextCKey);

return $nextCKey;

}

/\*\*

\* AuthModel::commuKeygen()

\*

\* 通訊 Key 產生器

\*

\* @param mixed $userId

\* @param mixed $gameId

\* @param mixed $roomId

\* @return

\*/

public function commuKeygen($userId, $gameId, $roomId)

{

$key = $this->keygen(10);

$cKey = sprintf("%s\_%d\_%d\_%d", $key, $userId, $gameId, $roomId);

return $cKey;

}

/\*\*

\* AuthModel::saveCommuKey()

\*

\* 儲存通訊 Key

\*

\* @param mixed $userId

\* @param mixed $gameId

\* @param mixed $key

\* @return void

\*/

public function saveCommuKey($userId, $gameId, $key)

{

$data = array('userId' => $userId, 'gameId' => $gameId, 'key' => $key);

$this->db->select("id");

$this->db->from('auth');

$this->db->where('userId', $userId);

$this->db->where('gameId', $gameId);

$result = $this->db->get()->result();

//echo count($result) > 0 ? "count > 0;" : "count <= 0";

if (count($result) > 0)

{

//echo $result[0]->id .";";

//print\_r($data);

$this->db->where('id', $result[0]->id);

$this->db->update('auth', $data);

//echo $this->db->last\_query();

}

else

{

$this->db->insert('auth', $data);

}

}

/\*\*

\* AuthModel::deleteCommuKey()

\*

\* 刪除通訊 Key

\*

\* @param mixed $cKey

\* @return void

\*/

public function deleteCommuKey($cKey)

{

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$this->db->where('userId', $userId);

$this->db->where('gameId', $gameId);

$this->db->delete('auth');

}

/\*\*

\* AuthModel::checkCommuKey()

\*

\* 檢查通訊 Key 是否存在

\*

\* @param mixed $cKey

\* @return

\*/

public function checkCommuKey($cKey)

{

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$this->db->select("key");

$this->db->from('auth');

$this->db->where('userId', $userId);

$this->db->where('gameId', $gameId);

$this->db->where('key', $cKey);

$result = $this->db->get()->result();

return count($result) > 0 ? true : false;

}

/\*\*

\* AuthModel::keygen()

\*

\* 金鑰產生器

\*

\* @param mixed $length

\* @return

\*/

public function keygen($length)

{

$key = '';

$microtime = microtime();

list($usec, $sec) = explode(' ', $microtime);

mt\_srand((float)$sec + ((float)$usec \* 100000));

$inputs = array\_merge(range('z', 'a'), range(0, 9), range('A', 'Z'));

for ($i = 0; $i < $length; $i++)

{

$key .= $inputs{mt\_rand(0, 61)};

}

return $key;

}

}

class ExecModel extends CI\_Model

{

/\*\*

\* ExecModel::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->database();

}

/\*\*

\* ExecModel::start()

\*

\* 遊戲開始

\*

\* @param mixed $userId

\* @param mixed $roomId

\* @param mixed $out

\* @param mixed $rooms

\* @return void

\*/

public function start($userId, $roomId, $out, $roomInfo, $roomPlayers)

{

//$this->db->trans\_begin();

if (count($roomInfo) <= 0)

{ // 房間不存在

//$this->db->trans\_rollback();

$out->wrong("Room isn't Exist");

}

if ($roomInfo[0]["max"] >= count($roomPlayers) && $roomInfo[0]["min"] <= count($roomPlayers))

{ // 確認房間內玩家 人數是否符合

if ($roomPlayers[0]["userId"] != $userId)

{ // 確認是否為室長

//$this->db->trans\_rollback();

$out->wrong("Participants Cannot Open Room");

}

// 寫入開房資料

$data = array("status" => "start", "turn" => $roomPlayers[0]["userId"],

"playingList" => $this->roomPlayersToString($roomPlayers));

$this->db->where("id", $roomId);

$this->db->update("gameroom", $data);

/\*if ($this->db->trans\_status() === false)

$this->db->trans\_rollback();

else

$this->db->trans\_commit();\*/

return $roomPlayers[0];

}

else

{

//$this->db->trans\_rollback();

$out->wrong("Opening Room Standard is Not Satisfied");

}

}

public function end($roomId)

{

$this->db->where("id", $roomId);

$data = array("arriveId" => null, "winList" => null, "playingList" => null,

"status" => "wait");

$this->db->update("gameroom", $data);

}

/\*\*

\* ExecModel::roomPlayersToString()

\*

\* 把 RoomPlayer 轉成 String

\*

\* @param mixed $roomPlayers

\* @return

\*/

private function roomPlayersToString($roomPlayers)

{

$ids = array();

foreach ($roomPlayers as $roomPlayer)

{

array\_push($ids, $roomPlayer["userId"]);

}

return implode("-", $ids);

}

/\*\*

\* ExecModel::send()

\*

\* 送事件至房間中的其他玩家

\*

\* @param mixed $message

\* @param mixed $senderId

\* @param mixed $roomId

\* @param mixed $roomPlayers

\* @return void

\*/

public function send($type, $param, $senderId, $roomId, $roomPlayers, $isIncludeSelf = false)

{

$insertDatas = array();

$insertLogs = array();

foreach ($roomPlayers as $roomPlayer)

{

if ($roomPlayer["userId"] != $senderId || $isIncludeSelf)

{

$data = array();

$data["type"] = $type;

$data["receiverId"] = $roomPlayer["userId"];

$data["roomId"] = $roomId;

$data["param"] = $param;

array\_push($insertDatas, $data);

$log = array();

$log["value"] = json\_encode($data);

$log["time"] = date("Y-m-d H:i:s");

array\_push($insertLogs, $log);

}

}

if (count($insertDatas) > 0)

{

$this->db->insert\_batch("event", $insertDatas);

$this->db->insert\_batch("log", $insertLogs);

}

}

public function listen($userId, $roomId, $out, $roomModel)

{

$this->db->select("id, type, param");

$this->db->from("event");

//$this->db->where("type", $type);

$this->db->where("receiverId", $userId);

$this->db->where("roomId", $roomId);

$this->db->order\_by("id", "ASC");

$result = $this->db->get()->result();

$array = array();

$lastEventId = 0;

foreach ($result as $row)

{

switch ($row->type)

{

case 'start':

case 'roomChanged':

$param = array("Players" => $roomModel->playerInfo($roomId, $out));

break;

default:

$param = $row->param;

}

array\_push($array, $out->convertToEvent($row->type, $param));

$lastEventId = $row->id;

}

// 刪除已讀訊息

if ($lastEventId != 0)

{

$this->db->where("receiverId", $userId);

$this->db->where("id <=", $lastEventId);

$this->db->delete("event");

}

return $array;

}

/\*\*

\* ExecModel::checkRoomIsStart()

\*

\* 確保此房間是 遊戲中 且存在

\*

\* @param mixed $roomInfos

\* @param mixed $out

\* @return void

\*/

public function checkRoomIsStart($roomInfos, $out)

{

if (count($roomInfos) <= 0)

{

$out->wrong("Cannot Send Message To Room");

}

}

/\*\*

\* ExecModel::next()

\*

\* 把回合控制器中的 turn 轉到下一位玩家

\*

\* @param mixed $roomInfos

\* @param mixed $userId

\* @param mixed $out

\* @return

\*/

public function next($roomInfo, $userId, $out)

{

// 計算下一位玩家

$turn = $roomInfo["turn"];

$list = explode("-", $roomInfo["list"]);

for ($i = 0; $i < count($list); $i++)

{

if ($list[$i] == $userId)

{

$nextPlayer = $list[($i + 1) % count($list)];

break;

}

}

// 更新資料庫

$data = array("turn" => $nextPlayer);

$this->db->where("id", $roomInfo["id"]);

$this->db->update("gameroom", $data);

return $nextPlayer;

}

/\*\*

\* ExecModel::removeFromPlayingList()

\*

\* 把自己從回合控制器中移除

\*

\* @param mixed $userId

\* @param mixed $roomInfo

\* @return void

\*/

public function removeFromPlayingList($userId, $roomInfo)

{

$roomId = $roomInfo["id"];

$list = explode("-", $roomInfo["list"]);

for ($i = 0; $i < count($list); $i++)

{

if ($list[$i] == $userId)

{

unset($list[$i]);

break;

}

}

$data = array("playingList" => implode("-", $list));

$this->db->where("id", $roomId);

$this->db->update("gameroom", $data);

}

}

class GameModel extends CI\_Model

{

/\*\*

\* GameModel::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->database();

}

/\*\*

\* GameModel::create()

\*

\* 創建遊戲

\*

\* @param mixed $name

\* @param mixed $gKey

\* @param mixed $password

\* @return

\*/

public function create($name, $gKey, $password)

{

$data = array('gameName' => $name, 'gKey' => $gKey, 'password' => $password);

$this->db->insert('gameinfo', $data);

return $gKey;

}

/\*\*

\* GameModel::exist()

\*

\* 確認某欄資料在 user資料表 中是否存在

\*

\* @param mixed $field

\* @param mixed $value

\* @return

\*/

public function exist($field, $value)

{

$this->db->select($field);

$this->db->from('gameinfo');

$this->db->where($field, $value);

return $this->db->count\_all\_results() > 0;

}

/\*\*

\* GameModel::getGameKey()

\*

\* 使用遊戲名稱 跟 開發者密碼 得到gKey

\*

\* @param mixed $name

\* @param mixed $password

\* @return

\*/

public function getGameKey($name, $password)

{

$this->db->select("gKey");

$this->db->from('gameinfo');

$this->db->where('gameName', $name);

$this->db->where('password', $password);

$result = $this->db->get()->result();

return count($result) > 0 ? $result[0]->gKey : 0;

}

/\*\*

\* GameModel::checkAuth()

\*

\* 確認 gKey

\*

\* @param mixed $gKey

\* @return

\*/

public function checkAuth($gKey)

{

$this->db->select("id, gameName");

$this->db->from('gameinfo');

$this->db->where('gKey', $gKey);

$result = $this->db->get()->result();

$isPermit = count($result) > 0;

return $isPermit ? array("id" => $result[0]->id, "name" => $result[0]->gameName) : false;

}

}

class GAuthModel extends CI\_Model

{

/\*\*

\* GAuthModel::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->database();

}

/\*\*

\* GAuthModel::saveLoginKey()

\*

\* 儲存登入 Key

\*

\* @param mixed $loginKey

\* @return void

\*/

public function saveLoginKey($loginKey)

{

list($key, $gameId) = explode('\_', $loginKey);

$data = array('gameId' => $gameId, 'key' => $loginKey);

$this->db->insert('gauth', $data);

}

/\*\*

\* GAuthModel::checkLoginKey()

\*

\* 確認登入 Key

\*

\* @param mixed $loginKey

\* @return

\*/

public function checkLoginKey($loginKey)

{

list($key, $gameId) = explode('\_', $loginKey);

$this->db->select("id");

$this->db->from('gauth');

$this->db->where('key', $loginKey);

$this->db->where('gameId', $gameId);

$result = $this->db->get()->result();

return count($result) > 0 ? true : false;

}

/\*\*

\* GAuthModel::deleteLoginKey()

\*

\* 刪除登入 Key

\*

\* @param mixed $loginKey

\* @return void

\*/

public function deleteLoginKey($loginKey)

{

list($key, $gameId) = explode('\_', $loginKey);

$this->db->where('gameId', $gameId);

$this->db->where('key', $loginKey);

$this->db->delete('gauth');

}

}

class OutputModel extends CI\_Model

{

private $output = array();

/\*\*

\* OutputModel::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

}

/\*\*

\* OutputModel::delete()

\*

\* 刪除儲存中的 key

\*

\* @param mixed $key

\* @return void

\*/

public function delete($key)

{

if (isset($this->output[$key]))

{

unset($this->output[$key]);

}

}

/\*\*

\* OutputModel::save()

\*

\* 儲存VALUE與KEY

\*

\* @param mixed $key

\* @param mixed $value

\* @return void

\*/

public function save($key, $value)

{

if ($value === true || $value === false)

$this->output[$key] = $value ? "true" : "false";

else

$this->output[$key] = $value;

}

/\*\*

\* OutputModel::debug()

\*

\* 給DEBUG用，以DEBUG KEY一直去累加要輸出的資料

\*

\* @param mixed $value

\* @return void

\*/

public function debug($value)

{

$key = "DebugInfo";

if (isset($this->output[$key]))

$this->output[$key] .= " ; " . $value;

else

$this->output[$key] = $value;

}

/\*\*

\* OutputModel::wrong()

\*

\* 重大錯誤，且危害程式執行，故停止執行程式

\*

\* @param mixed $value

\* @return void

\*/

public function wrong($value)

{

$this->save("Wrong", $value);

$this->show();

exit;

}

/\*\*

\* OutputModel::show()

\*

\* 把暫存資料以JSON格式輸出

\*

\* @return void

\*/

public function show()

{

echo json\_encode($this->output);

}

/\*\*

\* OutputModel::flush()

\*

\* 把暫存資料以JSON格式，透過SSE輸出

\*

\* @return void

\*/

public function flush()

{

echo "data: ";

$this->out->show();

echo "\n\n";

$this->output = array();

flush();

}

/\*\*

\* OutputModel::convertToEvent()

\*

\* Event Format is "array("Type" => $key, "Param" => $value)"

\*

\* @param mixed $key

\* @param mixed $value

\* @return

\*/

public function convertToEvent($key, $value)

{

return array("Type" => $key, "Param" => $value);

}

}

class RoomModel extends CI\_Model

{

/\*\*

\* RoomModel::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->database();

}

/\*\*

\* RoomModel::create()

\*

\* 創立房間

\*

\* @param mixed $gameId

\* @param mixed $title

\* @param mixed $minPlayer

\* @param mixed $maxPlayer

\* @return

\*/

public function create($gameId, $title, $minPlayer, $maxPlayer)

{

$data = array('gameId' => $gameId, 'title' => $title, 'min' => $minPlayer, 'max' =>

$maxPlayer, 'status' => 'wait');

$this->db->insert('gameroom', $data);

return $this->db->insert\_id();

}

/\*\*

\* RoomModel::join()

\*

\* 加入房間

\*

\* @param mixed $userId

\* @param mixed $roomId

\* @param mixed $out

\* @return

\*/

public function join($userId, $roomId, $out)

{

$this->checkUserNotInAnyRoom($userId, $out); // 確認玩家不要重複加入房間

$this->db->trans\_begin();

$this->checkRoomCanJoin($roomId, $out);

$data = array("roomId" => $roomId, "userId" => $userId);

$this->db->insert('room\_to\_user', $data);

if ($this->db->trans\_status() === false)

{

$this->db->trans\_rollback();

return false;

}

else

{

$this->db->trans\_commit();

return $roomId;

}

}

/\*\*

\* RoomModel::leave()

\*

\* 離開房間

\*

\* @param mixed $userId

\* @param mixed $roomId

\* @param mixed $out

\* @return void

\*/

public function leave($userId, $roomId, $out)

{

$this->db->where("roomId", $roomId);

$this->db->where("userId", $userId);

$this->db->delete('room\_to\_user');

$this->db->trans\_begin();

$plasyers = $this->getRoomPlayers($roomId);

if (count($plasyers) <= 0)

{

$this->db->where("id", $roomId);

$this->db->delete("gameroom");

}

if ($this->db->trans\_status() === false)

$this->db->trans\_rollback();

else

$this->db->trans\_commit();

}

/\*\*

\* RoomModel::roomInfo()

\*

\* 未開始房間資訊

\*

\* @param mixed $out

\* @param bool $roomId

\* @param string $status

\* @return

\*/

public function roomInfo($out, $roomId = false, $status = "wait")

{

$result = $this->getRooms($out, $roomId, $status);

$room = array();

foreach ($result as $row)

{

$array = array();

$array["id"] = $row->id;

$array["title"] = $row->title;

$array["max"] = $row->max;

$array["min"] = $row->min;

$array["now"] = $row->now == null ? 0 : $row->now;

$array["turn"] = $row->turn;

$array["list"] = $row->playingList;

if ($row->id != null)

array\_push($room, $array);

}

$out->save("Room", $room);

return $room;

}

/\*\*

\* RoomModel::playerInfo()

\*

\* 房間中玩家資訊

\*

\* @param mixed $roomId

\* @param mixed $out

\* @return

\*/

public function playerInfo($roomId, $out)

{

$result = $this->getRoomPlayers($roomId);

$players = array();

foreach ($result as $row)

{

$array = array();

$array["userId"] = $row->id;

$array["userName"] = $row->userName;

array\_push($players, $array);

}

$out->save("Players", $players);

return $players;

}

/\*\*

\* RoomModel::getRooms()

\*

\* 得到未開始房間(未處理成array物件)

\*

\* @param mixed $out

\* @param mixed $roomId

\* @param mixed $status

\* @return

\*/

private function getRooms($out, $roomId, $status)

{

// 各房間中有多少人的 Table(RoomId, NowPlayers)

$RoomPlayerCountTable =

"(SELECT roomId, count(userId) AS now from room\_to\_user GROUP BY roomId) AS RPCT";

$this->db->select("gameroom.id, title, min, max, turn, now, playingList");

$this->db->from("gameroom");

if ($roomId != false)

$this->db->where("gameroom.id", $roomId);

$this->db->where("status", $status);

$this->db->join($RoomPlayerCountTable, "gameroom.id = RPCT.roomId", "left");

return $this->db->get()->result();

}

/\*\*

\* RoomModel::getRoomPlayers()

\*

\* 得到房間中玩家資訊(未處理成array物件)

\*

\* @param mixed $roomId

\* @return

\*/

private function getRoomPlayers($roomId)

{

$this->db->select("user.id, user.userName");

$this->db->from("room\_to\_user");

$this->db->where("room\_to\_user.roomId", $roomId);

$this->db->join("user", "user.id = room\_to\_user.userId", "left");

$this->db->order\_by("room\_to\_user.id", "ASC");

return $this->db->get()->result();

}

/\*\*

\* RoomModel::modify()

\*

\* 修改房間資訊

\*

\* @param mixed $roomId

\* @param mixed $data

\* @return void

\*/

public function modify($roomId, $data)

{

$this->db->where("id", $roomId);

$this->db->update('gameroom', $data);

}

/\*\*

\* RoomModel::deleteSelfFromAnyRoom()

\*

\* 刪除所有自己所在之任何房間

\*

\* @param mixed $userId

\* @return void

\*/

public function deleteSelfFromAnyRoom($userId)

{

$this->db->where("userId", $userId);

$this->db->delete("room\_to\_user");

}

/\*\*

\* RoomModel::checkRoomCanJoin()

\*

\* 確認此房間是否能加入

\*

\* @param mixed $roomId

\* @param mixed $out

\* @return void

\*/

public function checkRoomCanJoin($roomId, $out)

{

$this->db->select("max");

$this->db->from('gameroom');

$this->db->where('id', $roomId);

$this->db->where('status', 'wait');

$result = $this->db->get()->result();

if (count($result) <= 0)

{

$this->db->trans\_rollback();

$out->wrong("No This Room or Status isn't Wait");

}

$max = $result[0]->max;

$this->db->select("roomId");

$this->db->from('room\_to\_user');

$this->db->where('roomId', $roomId);

$result = $this->db->get()->result();

if (count($result) + 1 > $max)

{

$this->db->trans\_rollback();

$out->wrong("This Room is Full");

}

}

/\*\*

\* RoomModel::checkUserNotInAnyRoom()

\*

\* 確認玩家有無在其他房間內

\*

\* @param mixed $userId

\* @param mixed $out

\* @return void

\*/

public function checkUserNotInAnyRoom($userId, $out)

{

$this->db->select("roomId");

$this->db->from('room\_to\_user');

$this->db->where('userId', $userId);

$result = $this->db->get()->result();

if (count($result) > 0)

{

$out->wrong(sprintf("User In %d Room", $result[0]->roomId));

}

}

public function waitCheckWin($roomId, $userId)

{

$this->db->where("id", $roomId);

$data = array("arriveId" => $userId);

$this->db->update("gameroom", $data);

}

public function getWaitCheckWinUserId($roomId)

{

$this->db->select("arriveId");

$this->db->from("gameroom");

$this->db->where("id", $roomId);

$result = $this->db->get()->result();

return $result[0]->arriveId;

}

public function saveToWinList($roomId, $userId)

{

$this->db->select("winList");

$this->db->from("gameroom");

$this->db->where("id", $roomId);

$result = $this->db->get()->result();

$winList = $result[0]->winList == null ? $userId : $result[0]->winList . "-" . $userId;

$this->db->where("id", $roomId);

$data = array("arriveId" => null, "winList" => $winList);

$this->db->update("gameroom", $data);

return $winList;

}

}

class UserModel extends CI\_Model

{

/\*\*

\* UserModel::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->database();

}

/\*\*

\* UserModel::create()

\*

\* 創建使用者

\*

\* @param mixed $name

\* @param mixed $account

\* @param mixed $password

\* @return

\*/

public function create($name, $account, $password)

{

$data = array('userName' => $name, 'account' => $account, 'password' => $password);

$this->db->insert('user', $data);

return $this->db->insert\_id();

}

/\*\*

\* UserModel::exist()

\*

\* 確認某欄資料在 user資料表 中是否存在

\*

\* @param mixed $field

\* @param mixed $value

\* @return

\*/

public function exist($field, $value)

{

$this->db->select($field);

$this->db->from('user');

$this->db->where($field, $value);

return $this->db->count\_all\_results() > 0;

}

/\*\*

\* UserModel::checkAuth()

\*

\* 確認使用者帳號密碼

\*

\* @param mixed $account

\* @param mixed $password

\* @return

\*/

public function checkAuth($account, $password)

{

$this->db->select("id, userName");

$this->db->from('user');

$this->db->where('account', $account);

$this->db->where('password', $password);

$result = $this->db->get()->result();

$isPermit = count($result) > 0;

return $isPermit ? array("id" => $result[0]->id, "name" => $result[0]->userName) : false;

}

}

* EzWebGame Controller

class Developers extends CI\_Controller

{

public function \_\_construct()

{

parent::\_\_construct();

$this->load->model("GameModel");

}

public function index()

{

$this->load->view('header');

$this->load->view('developers');

$this->load->view('footer');

}

}

class Event extends CI\_Controller

{

public function \_\_construct()

{

parent::\_\_construct();

$this->load->model("OutputModel", "out");

$this->load->model("ExecModel");

$this->load->model("AuthModel");

}

public function request($cKey)

{

$this->checkIsPermit($cKey);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$this->load->model("RoomModel", "room");

header("Content-Type: text/event-stream");

header('Cache-Control: no-cache');

if (ob\_get\_level())

ob\_end\_flush();

$this->out->save("Events", $this->out->convertToEvent("Debug", "Hello Request"));

$this->out->flush();

$count = 0;

while (true)

{

if ($count++ >= 3)

{// 過一陣子 要回傳一些訊息給 SSE

$this->out->save("Events",$this->out->convertToEvent("Debug", "Test Request"));

$this->out->flush();

$count = 0;

}

if ($roomId != 0)

{

$events = $this->ExecModel->listen($userId, $roomId, $this->out, $this->room);

if (count($events) > 0)

{

$this->out->save("Events", $events);

$this->out->flush();

$count = 0;

}

}

else

{

$rooms = $this->room->roomInfo($this->out);

$this->out->save("Events", array($this->out->convertToEvent("RefreshRoomList", $rooms)));

$this->out->flush();

$count = 0;

}

sleep(3);

}

}

private function checkIsPermit($cKey)

{

$isPermit = $this->AuthModel->checkCommuKey($cKey);

if (!$isPermit)

{ // 通訊Key 認證失敗

$this->out->wrong("No Auth Can Request");

}

}

}

class Exec extends CI\_Controller

{

public function \_\_construct()

{

parent::\_\_construct();

$this->load->model("OutputModel", "out");

$this->load->model("ExecModel");

$this->load->model("AuthModel");

}

/\*\*

\* Exec::start()

\*

\* 開始遊戲

\*

\* @param mixed $cKey

\* @return void

\*/

public function start($cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

// 獲取房間資料

$this->load->model("RoomModel", "room");

$roomInfo = $this->room->roomInfo($this->out, $roomId);

$this->out->delete("Room");

$roomPlayers = $this->room->playerInfo($roomId, $this->out);

$turnPlayer = $this->ExecModel->start($userId, $roomId, $this->out, $roomInfo, $roomPlayers);

// 告知其他玩家 遊戲開始

$message = sprintf("Room[%d] Start Game", $roomId);

$this->ExecModel->send("start", $message, $userId, $roomId, $roomPlayers);

// 告知所有玩家 現在換誰

$message = json\_encode($turnPlayer);

$this->ExecModel->send("turn", $message, $userId, $roomId, $roomPlayers, true);

$this->out->save("Start", true);

$this->out->show();

}

/\*\*

\* Exec::SendMessage()

\*

\* 遊戲中 傳送指令給其他玩家

\*

\* @param mixed $message

\* @param mixed $cKey

\* @return void

\*/

public function SendMessage($cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$message = $this->input->post('message');

$roomInfo = $this->checkRoomExistAndIsTurnMe($userId, $roomId);

$roomPlayers = $this->room->playerInfo($roomId, $this->out);

$this->ExecModel->send("message", $message, $userId, $roomId, $roomPlayers);

$this->out->save("Message", $message);

$this->out->show();

}

/\*\*

\* Exec::nextRound()

\*

\* @param mixed $cKey

\* @return void

\*/

public function nextRound($cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$roomInfo = $this->checkRoomExistAndIsTurnMe($userId, $roomId);

$playerId = $this->ExecModel->next($roomInfo, $userId, $roomId);

$roomPlayers = $this->room->playerInfo($roomId, $this->out);

foreach ($roomPlayers as $player)

{

if ($player["userId"] == $playerId)

{

$this->out->save("NextRound", $player);

// 告知其他玩家 現在換誰

$message = json\_encode($player);

$this->ExecModel->send("turn", $message, $userId, $roomId, $roomPlayers);

break;

}

}

$this->out->show();

}

/\*\*

\* Exec::ArriveFinalStep()

\*

\* 找個玩家 說你已經獲勝 看他同不同意

\*

\* @param mixed $cKey

\* @return void

\*/

public function ArriveFinalStep($cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$roomInfo = $this->checkRoomExistAndIsTurnMe($userId, $roomId);

//尋找適合的玩家 去確認是否此人完成遊戲

$askPlayer = array();

$roomPlayers = $this->room->playerInfo($roomId, $this->out);

foreach($roomPlayers as $player)

{

if($player["userId"] != $userId)

{

array\_push($askPlayer, $player);

break;

}

}

$this->room->waitCheckWin($roomId, $userId);

$message = json\_encode(array("WinnerId" => $userId));

$this->ExecModel->send("checkWin", $message, $userId, $roomId, $askPlayer);

$this->out->show();

}

/\*\*

\* Exec::Replay()

\*

\* 回復 剛剛說WIN的玩家是否獲勝

\*

\* @param mixed $isWin

\* @param mixed $cKey

\* @return void

\*/

public function Reply($isWin, $cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$this->load->model("RoomModel", "room");

$roomPlayers = $this->room->playerInfo($roomId, $this->out);

$winUserId = $this->room->getWaitCheckWinUserId($roomId);

if($isWin == "true")

{

$roomInfos = $this->room->roomInfo($this->out, $roomId, "start");

$roomInfo = $roomInfos[0];

$playerId = $this->ExecModel->next($roomInfo, $winUserId, $roomId);

$this->ExecModel->removeFromPlayingList($winUserId, $roomInfo);

foreach ($roomPlayers as $player)

{

if ($player["userId"] == $playerId)

{

$turnWhoMessage = json\_encode($player);

}

if($player["userId"] == $winUserId)

{

$whoArrived = json\_encode($player);

}

}

// 把玩家存入 winList

$winList = $this->room->saveToWinList($roomId, $winUserId);

// 確認此房間是否已經結束

$roomInfos = $this->room->roomInfo($this->out, $roomId, "start");

$roomInfo = $roomInfos[0];

if(count(explode("-", $roomInfo["list"])) == 1)

{

$rank = explode("-", $winList."-".$roomInfo["list"]);

foreach ($rank as &$Id)

{

foreach ($roomPlayers as $player)

{

if ($player["userId"] == $Id)

{

$Id = $player;

break;

}

}

}

// 把房間改成wait

$this->ExecModel->end($roomId);

// 告知所有人 Rank And GameOver

$this->ExecModel->send("rank", json\_encode($rank), $userId, $roomId, $roomPlayers, true);

}

else

{

// 告知有玩家抵達終點

$this->ExecModel->send("arrived", $whoArrived, $userId, $roomId, $roomPlayers, true);

// 告知所有玩家 現在換誰

$this->ExecModel->send("turn", $turnWhoMessage, $userId, $roomId, $roomPlayers, true);

}

}

else

{

$message = "Cheat";

$this->ExecModel->send("arrived", $message, $userId, $roomId, $roomPlayers, true);

}

$this->out->show();

}

/\*\*

\* Exec::checkRoomExistAndIsTurnMe()

\*

\* @param mixed $userId

\* @param mixed $roomId

\* @return 傳回目前房間資訊

\*/

private function checkRoomExistAndIsTurnMe($userId, $roomId)

{

// 確認房間

$this->load->model("RoomModel", "room");

$roomInfos = $this->room->roomInfo($this->out, $roomId, "start");

$this->out->delete("Room"); // 刪除儲存在out中的Room Key

$this->ExecModel->checkRoomIsStart($roomInfos, $this->out);

// 確認現在是輪到自己送訊息

if ($roomInfos[0]["turn"] == $userId)

{

return $roomInfos[0];

}

else

{

$this->out->wrong("Game isn't turn me");

}

}

}

class Game extends CI\_Controller

{

/\*\*

\* Game::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->model("GameModel");

}

/\*\*

\* Game::create()

\*

\* 創建遊戲

\*

\* @param mixed $name

\* @param mixed $password

\* @return void

\*/

public function create($name, $password)

{

if (!$this->isNameExist($name))

{

$this->load->model("AuthModel");

$gKey = $this->AuthModel->keygen(18);

echo 'Your Key is<br>' . $this->GameModel->create($name, $gKey, $password);

}

else

{

echo "Name has been used!";

}

}

/\*\*

\* Game::isNameExist()

\*

\* 此遊戲名稱是否存在

\*

\* @param mixed $name

\* @return

\*/

public function isNameExist($name)

{

return $this->GameModel->exist("gameName", $name);

}

/\*\*

\* Game::getGameKey()

\*

\* 得到 gKey

\*

\* @param mixed $name

\* @param mixed $password

\* @return void

\*/

public function getGameKey($name, $password)

{

$gKey = $this->GameModel->getGameKey($name, $password);

if ($gKey == '0')

echo 'Incorrect Name or Password';

else

echo 'Your Key is<br>' . $gKey;

}

/\*\*

\* Game::loadEzWebGameLib()

\*

\* 下載 EzWebGameLib

\*

\* @param mixed $gKey

\* @return void

\*/

public function loadEzWebGameLib($gKey)

{

$auth = $this->GameModel->checkAuth($gKey);

//print\_r($auth);

if ($auth != false)

{

$this->load->model("AuthModel");

$this->load->model("GAuthModel");

$loginKey = sprintf("%s\_%d", $this->AuthModel->keygen(12), $auth["id"]);

$this->GAuthModel->saveLoginKey($loginKey);

echo $loginKey;

}

else

{

echo "404 Error";

}

}

}

class Member extends CI\_Controller

{

public function \_\_construct()

{

parent::\_\_construct();

$this->load->model("GameModel");

}

public function index()

{

$this->signUP();

}

public function signUP()

{

$this->load->view('header');

$this->load->view('Member/signup');

$this->load->view('footer');

}

public function login($lKey = null)

{

$this->load->library('user\_agent');

$data["lKey"] = $lKey;

$this->load->view('header');

if ($this->agent->is\_referral())

{

$this->load->view('Member/login', $data);

}

$this->load->view('footer');

}

}

class Room extends CI\_Controller

{

/\*\*

\* Room::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->model("OutputModel", "out");

$this->load->model("RoomModel");

$this->load->model("AuthModel");

}

/\*\*

\* Room::create()

\*

\* 創建房間

\*

\* @param mixed $title

\* @param mixed $minPlayer

\* @param mixed $maxPlayer

\* @param mixed $cKey

\* @return void

\*/

public function create($title, $minPlayer, $maxPlayer, $cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

$this->checkPlayerNumber($minPlayer, $maxPlayer);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$this->RoomModel->checkUserNotInAnyRoom($userId, $this->out); //檢查玩家是否有在任何房間中

$roomId = $this->RoomModel->create($gameId, $title, $minPlayer, $maxPlayer);

$this->out->save("Create", $roomId);

$this->join($roomId, $nextCKey); //自動加入剛創建的房間

}

/\*\*

\* Room::join()

\*

\* 加入房間

\*

\* @param mixed $iRoomId

\* @param mixed $cKey

\* @return void

\*/

public function join($iRoomId, $cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$roomId = $this->RoomModel->join($userId, $iRoomId, $this->out);

if ($roomId != false)

{

$this->RoomModel->roomInfo($this->out, $roomId);

$roomPlayers = $this->RoomModel->playerInfo($roomId, $this->out);

// 告知其他玩家 自己加入房間

$this->load->model("ExecModel", "Exec");

$message = sprintf("User[%d] Join Room[%d]", $userId, $roomId);

$this->Exec->send("roomChanged", $message, $userId, $roomId, $roomPlayers);

}

$this->AuthModel->editCommuKey($nextCKey, $iRoomId, $this->out);

$this->out->save("Join", $roomId);

$this->out->show();

}

/\*\*

\* Room::leave()

\*

\* 離開房間

\*

\* @param mixed $cKey

\* @return void

\*/

public function leave($cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$this->RoomModel->leave($userId, $roomId, $this->out);

$this->AuthModel->editCommuKey($nextCKey, 0, $this->out);

// 告知其他玩家 自己離開房間

$this->load->model("ExecModel", "Exec");

$roomPlayers = $this->RoomModel->playerInfo($roomId, $this->out);

$this->out->delete("Players");

$message = sprintf("User[%d] Leave Room[%d]", $userId, $roomId);

$this->Exec->send("roomChanged", $message, $userId, $roomId, $roomPlayers);

$this->out->save("Leave", true);

$this->out->show();

}

/\*\*

\* Room::ListRoomInfos()

\*

\* 查詢所有未開始房間

\*

\* @param mixed $cKey

\* @return void

\*/

public function ListRoomInfos($cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$this->RoomModel->roomInfo($this->out);

$this->out->show();

}

/\*\*

\* Room::ListRoomPlayers()

\*

\* 查詢某房間的所有玩家

\*

\* @param mixed $cKey

\* @return void

\*/

public function ListRoomPlayers($cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$this->RoomModel->playerInfo($roomId, $this->out);

$this->out->show();

}

/\*\*

\* Room::modifyTitle()

\*

\* 修改房間名稱

\*

\* @param mixed $iTitle

\* @param mixed $cKey

\* @return void

\*/

public function modifyTitle($iTitle, $cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

$data = array("title" => $iTitle);

$this->RoomModel->modify($roomId, $data);

$this->out->save("ModifyTitle", true);

}

/\*\*

\* Room::ModifyMinMaxPlayer()

\*

\* 修改房間 玩家人數上下限

\*

\* @param mixed $minPlayer

\* @param mixed $maxPlayer

\* @param mixed $cKey

\* @return void

\*/

public function ModifyMinMaxPlayer($minPlayer, $maxPlayer, $cKey)

{

$nextCKey = $this->AuthModel->getNextCommuKey($cKey, $this->out);

list($key, $userId, $gameId, $roomId) = explode('\_', $cKey);

checkPlayerNumber($minPlayer, $maxPlayer);

$data = array("min" => $minPlayer, "max" => $maxPlayer);

$this->RoomModel->modify($roomId, $data);

$this->out->save("ModifyTitle", true);

}

/\*\*

\* Room::checkPlayerNumber()

\*

\* 遊戲設定玩家人數上下限 防呆機制

\*

\* @param mixed $minPlayer

\* @param mixed $maxPlayer

\* @return void

\*/

private function checkPlayerNumber($minPlayer, $maxPlayer)

{

if (!($minPlayer >= 1 && $minPlayer <= $maxPlayer && $maxPlayer <= 10))

{

$this->out->wrong("MaxPlayer = 2~20, MinPlayer <= 2 <=MaxPlayer");

}

}

}

class User extends CI\_Controller

{

/\*\*

\* User::\_\_construct()

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

$this->load->model("OutputModel", "out");

$this->load->model("UserModel");

}

/\*\*

\* User::signup()

\*

\* 創建使用者帳號密碼

\*

\* @param mixed $name

\* @param mixed $account

\* @param mixed $password

\* @return void

\*/

public function signup($name, $account, $password)

{

if (!$this->UserModel->exist("account", $account) && !$this->UserModel->exist("userName",

$name))

{

$userId = $this->UserModel->create($name, $account, $password);

$this->out->save("userId", $userId);

}

else

{

$this->out->wrong("Account or UserName Repeat");

}

$this->out->show();

}

/\*\*

\* User::isNameExist()

\*

\* 確認名字是否存在

\*

\* @param mixed $name

\* @return void

\*/

public function isNameExist($name)

{

$isExist = $this->UserModel->exist("userName", $name);

$this->out->save("NameExist", $isExist);

$this->out->show();

}

/\*\*

\* User::isAccountExist()

\*

\* 確認此帳號是否存在

\*

\* @param mixed $account

\* @return void

\*/

public function isAccountExist($account)

{

$isExist = $this->UserModel->exist("account", $account);

$this->out->save("AccountExist", $isExist);

$this->out->show();

}

/\*\*

\* User::login()

\*

\* 登入

\*

\* @param mixed $lKey

\* @param mixed $account

\* @param mixed $password

\* @return void

\*/

public function login($lKey, $account, $password)

{

$this->load->model("GAuthModel");

$gAuth = $this->GAuthModel->checkLoginKey($lKey); // 確認此key可以用來登入此遊戲

$auth = $this->UserModel->checkAuth($account, $password);

if ($gAuth && $auth != false)

{

$this->load->model("AuthModel");

$this->GAuthModel->deleteLoginKey($lKey); // 刪除登入時使用的Key

list($key, $gameId) = explode('\_', $lKey);

$nextCKey = $this->AuthModel->commuKeygen($auth["id"], $gameId, 0); // 產生 溝通key

$this->AuthModel->saveCommuKey($auth["id"], $gameId, $nextCKey); // 儲存溝通key

$this->out->save("cKey", $nextCKey);

// 刪除之前所在之房間

$this->load->model("RoomModel", "room");

$this->room->deleteSelfFromAnyRoom($auth["id"]);

}

else

{

$this->out->wrong("Authentication failed");

}

$this->out->show();

}

/\*\*

\* User::cancelLogin()

\*

\* 取消此 登入Key

\*

\* @param mixed $lKey

\* @return void

\*/

public function cancelLogin($lKey)

{

$this->load->model("GAuthModel");

$this->GAuthModel->deleteLoginKey($lKey);

}

/\*\*

\* User::logout()

\*

\* 登出

\*

\* @param mixed $ckey

\* @return void

\*/

public function logout($ckey)

{

$this->load->model("AuthModel");

$this->AuthModel->deleteCommuKey($ckey);

$this->out->save("Logout", true);

$this->out->show();

}

}

* EzWebGame View

header.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<meta name="description" content="" />

<meta name="author" content="keming&ZhongTing" />

<link rel="shortcut icon" href="<?=base\_url("assets/icon/favicon.ico")?>" />

<title>EzWebGames</title>

<!-- Bootstrap core CSS -->

<link href="<?=base\_url("assets/css/bootstrap.css")?>" rel="stylesheet" />

<!-- Custom styles for this template -->

<link href="<?=base\_url("assets/css/jumbotron-narrow.css")?>" rel="stylesheet" />

<!-- HTML5 shim and Respond.js IE8 support of HTML5 elements and media queries -->

<!--[if lt IE 9]>

<script src="../../assets/js/html5shiv.js"></script>

<script src="../../assets/js/respond.min.js"></script>

<![endif]-->

<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>

</head>

<body>

<div class="container">

<div class="header">

<ul class="nav nav-pills pull-right">

<li class="active"><a href="<?=base\_url("")?>">Home</a></li>

<li><a href="<?=base\_url("Developers")?>">Developers</a></li>

<li><a href="<?=base\_url("Member")?>">Member</a></li>

</ul>

<h3 class="text-muted">EzWebGame</h3>

</div>

<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>

<h4 class="modal-title">Modal title</h4>

</div>

<div class="modal-body">

...

</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal">OK</button>

</div>

</div><!-- /.modal-content -->

</div><!-- /.modal-dialog -->

</div>

index.php

<div class="jumbotron">

<h1>EzWebGame</h1>

<p class="lead">Enjoy Games in EzWebGame</p>

<p><a class="btn btn-lg btn-success" href="<?=base\_url('member/signup')?>">Sign up today</a></p>

</div>

<div class="row marketing">

<div class="col-lg-6">

<h4>Developer</h4>

<p>Register your game and get EzWebGameLib for free!</p>

</div>

<div class="col-lg-6">

<h4>Member</h4>

<p>Become a member and play games on EzWebGames!</p>

</div>

</div>

developers.php

<input type="button" class="btn btn-primary" value="Register Game"/>

<div id="dataInputDIV">

<br />

<form role="form">

<div class="form-group">

<label for="exampleInputEmail1">Game Name</label>

<input id="gameName" type="text" class="form-control" placeholder="Enter your game name">

</div>

<div class="form-group">

<label for="exampleInputPassword1">Password</label>

<input id="password" type="password" class="form-control" placeholder="Password">

</div>

<input type="button" class="btn btn-default" value="Register"/>

<input type="button" class="btn btn-default" value="Query Key"/>

</form>

</div>

<br />

<br />

<input type="button" class="btn btn-primary" value="Query GameKey"/>

<br />

<br />

<input type="button" class="btn btn-primary" value="Download Library"/>

<br />

<br />

<script type="text/javascript">

$(".header ul li").each(function(){$(this).removeClass('active')});

$(".header ul li:eq(1)").addClass('active');

$("#dataInputDIV").hide();

$("input[value='Register Game']").on('click',function(){

if($(dataInputDIV).css('display')=='none')

$("#dataInputDIV").slideToggle();

else if($("#gameName").val().trim()!="")

sendNameAndPasswordTo('./game/create','Register Result');

$("#dataInputDIV input[value='Register']").show();

$("#dataInputDIV input[value='Query Key']").hide();

});

$("input[value='Query GameKey']").on('click',function(){

if($(dataInputDIV).css('display')=='none')

$("#dataInputDIV").slideToggle();

else if($("#gameName").val().trim()!="")

sendNameAndPasswordTo('./game/getGameKey','Query Result');

$("#dataInputDIV input[value='Register']").hide();

$("#dataInputDIV input[value='Query Key']").show();

});

$("input[value='Download Library']").on('click',function(){

window.open("http://140.124.181.7:1221/GameRound/EzWebGameLib.rar");

});

$("#dataInputDIV input[value='Register']").on('click',function(){

sendNameAndPasswordTo('./game/create','Register Result');

});

$("#dataInputDIV input[value='Query Key']").on('click',function(){

sendNameAndPasswordTo('./game/getGameKey','Query Result');

});

function sendNameAndPasswordTo(baseURL,resultModalTitle)

{

var name = $("#gameName").val().trim();

var password = $("#password").val().trim();

if(name==""||password=="")

{

$('#myModal .modal-title').text('Error');

$('#myModal .modal-body').html("Please complete game name and password");

$('#myModal').modal();

}

else

{

$.ajax({

url: baseURL+'/'+name+'/'+password

}).done(function(data) {

$('#myModal .modal-title').text(resultModalTitle);

$('#myModal .modal-body').html(data);

$('#myModal').modal();

});

}

}

</script>

Member/login.php

<span class="label label-primary">&nbsp Account&nbsp</span>

<input type="text" id="account" class="form-control" placeholder="Enter your account">

<br />

<span class="label label-primary">Password</span>

<input type="password" id="password" class="form-control" placeholder="Enter your password">

<br />

<input id="login" type="button" class="btn btn-primary" value="Login"/>

<script>

window.onblur = window.close;

window.onbeforeunload = function(){$.ajax({url: "<?=base\_url('/user/cancelLogin/'.$lKey)?>",})};

(function(){

$("#account").focus();

if(opener != null && opener.EzWebGame!=null)

{

$("#login").click(function(){

var account = $("#account").val();

var password = $("#password").val();

if(account==""||password=="")

{

alert("account or password empty");

return false;

}

$.ajax({

url: "<?=base\_url('/user/login/')?>"+"/<?php echo $lKey?>"+'/'+account+'/'+password,

}).done(function(data) {

window.close();

data = JSON.parse(data);

opener.EzWebGame.cKey(data);

});

});

}

else

{

$("#login").click(function(){

alert('coming soon');

});

}

})()

</script>

Member/signup.php

<span class="label label-primary">&nbsp&nbsp&nbsp Name&nbsp&nbsp&nbsp</span>

<input type="text" id="name" class="form-control" placeholder="Enter your name">

<br />

<span class="label label-primary">&nbsp Account&nbsp</span>

<input type="text" id="account" class="form-control" placeholder="Enter your account">

<br />

<span class="label label-primary">Password</span>

<input type="password" id="password" class="form-control" placeholder="Enter your password">

<br />

<input type="button" class="btn btn-primary" value="Sign UP!"/>

<br />

<br />

<script>

$(".header ul li").each(function(){$(this).removeClass('active')});

$(".header ul li:eq(2)").addClass('active');

$("input[value='Sign UP!']").click(function(){

var name = $("#name").val().trim();

var account = $("#account").val().trim();

var password = $("#password").val().trim();

if(name==""||account==""||password=="")

{

$('#myModal .modal-title').text('Error');

$('#myModal .modal-body').html("Please complete game name, account, and password");

$('#myModal').modal();

}

else

{

$.ajax({

url: '../user/signup'+'/'+name+'/'+account+'/'+password

}).done(function(data) {

$('#myModal .modal-title').text('SignUp Result');

$('#myModal .modal-body').html(data);

$('#myModal').modal();

});

}

});

</script>

* EzWebChecker Script

script/canvas.js

turnToLoginLayer();

//turnToRoomLayer();

//turnToGameLayer();

function turnToLoginLayer()

{

initLayer();

loginLayer.show().draw();

}

function initLayer()

{

stage.getLayers().each(function(layer){layer.hide()});

backgroundLayer.show();

}

function turnToLobbyLayer()

{

initLayer();

lobbyLayer.show().draw();

}

function turnToRoomLayer()

{

initLayer();

roomLayer.show().draw();

}

function turnToGameLayer()

{

initLayer();

chessBoardLayer.show().draw();

gameLayer.show().draw();

gameEffectLayer.show().draw();

}

function refreshRoomInfoLayer(room)

{

roomInfoLayer.removeChildren();

roomInfoLayer.add(newLabel(0,10,room.title,stage.getWidth()-10,30));

roomInfoLayer.add(newLabel(stage.getWidth()-210,60,'MaxPlayer: '+room.max,200,20));

roomInfoLayer.show().draw();

}

function refreshPlayersInRoomInfoLayer(players)

{

var posX = [100,260,420];

var pics = ['./red.jpg','./yellow.jpg','green.jpg'];

if(roomPlayerLayer)roomPlayerLayer.clear();

roomPlayerLayer = new Kinetic.Layer();

for(var i=0; i < players.length; i++)

{

newPlayerZone(posX[i],150,150,150,players[i].userName,pics[i],roomPlayerLayer);

}

//roomPlayerLayer.show().draw();

stage.add(roomPlayerLayer);

}

function refreshLobbyRooms(roomInfos)

{

var x = 100,y=100,width=stage.getWidth()-y-10,height=30;

if(lobbyRoomsLayer)lobbyRoomsLayer.clear();

lobbyRoomsLayer = new Kinetic.Layer();

for(var i =0;i<roomInfos.length;i++)

{

newLobbyRoomZone(x,y,width,height,roomInfos[i],lobbyRoomsLayer);

y+=50;

}

stage.add(lobbyRoomsLayer);

}

function clickChecker(point)

{

if(point.player>=0 && EzWebGame.isTurnSelf() && EzWebGame.getUserTurnOrder() == point.player)

{

EzWebGame.doStep(JSON.stringify({"Method":"Select", "Point":{x:point.x,y:point.y}}));

selectedChecker = point;

displaySelectCheckerEffect(point);

displayPlaceToMove(point);

}

}

function displayPlaceToMove(point)

{

findAndRecordOnBoard(point);

showPlaceToMoveEffect();

}

function showPlaceToMoveEffect()

{

var moveDirection = getMoveDirection();

for(var i in chessPoints)

{

if(!chessPoints[i].computed)continue;

var p = gridXyToXy(chessPoints[i]);

var c = new Kinetic.Circle({

x: p.x,

y: p.y,

radius: 15,

fill: 'white',

stroke: 'black',

//strokeWidth: 1,

});

c.attrs.point = chessPoints[i];

c.on('mouseover',function(evt){

showPreviewPath(selectedChecker,evt.targetNode.attrs.point);

});

c.on('mouseout',function(){

gameEffectLayer.find('#previewLine').remove();

gameEffectLayer.clear().draw();

})

if(EzWebGame.isTurnSelf())

{

c.on('click',function(evt){

var point = evt.targetNode.attrs.point;

EzWebGame.doStep(JSON.stringify({"Method":"MoveTo", "Point":{x:point.x,y:point.y}}));

moveCheckerTo(point,function(){

//判定獲勝

if(isWin(EzWebGame.getUserTurnOrder(EzWebGame.getUserId())))

{//告知其他人 自己獲勝遊戲

EzWebGame.finishGame();

}

else

{//結束回合

EzWebGame.finishStep();

}

});

});

}

gameEffectLayer.add(c);

delete chessPoints[i].computed;

}

gameEffectLayer.clear();

gameEffectLayer.draw();

}

function showPreviewPath(startPoint,finalPoint)

{

var path = findPath(startPoint,finalPoint);

var points = getLinePointsArray(path);

var line = new Kinetic.Line({

points: points,

stroke: userCheckerColors[EzWebGame.getNowTurnUserOrder()],

strokeWidth: 5,

lineCap: 'round',

lineJoin: 'round',

dashArray: [10, 10],

id:'previewLine'

});

gameEffectLayer.find('#previewLine').remove();

gameEffectLayer.add(line);

gameEffectLayer.clear().draw();

}

function getLinePointsArray(arrayPoint)

{

var points = [];

for(var i =0;i<arrayPoint.length;i++)

{

var p = gridXyToXy(arrayPoint[i]);

points.push(p.x);

points.push(p.y);

}

return points;

}

function displaySelectCheckerEffect(point)

{

function cancel(){

EzWebGame.doStep(JSON.stringify({"Method":"CancelSelect", "Point":{x:point.x,y:point.y}}));

}

greyBackgroundEffect(gameEffectLayer,cancel,194);

gameEffectLayer.add(new Kinetic.Circle(point.circle));

gameEffectLayer.clear().draw();

}

function moveCheckerTo(point,callBack)

{

gameEffectLayer.removeChildren();

var path = findPath(selectedChecker,point);

var points = getLinePointsArray(path);

var line = new Kinetic.Line({

points: points,

//stroke: userCheckerColors[selectedChecker.player],

stroke:'black',

strokeWidth: 5,

lineCap: 'round',

lineJoin: 'round',

id:'pathLine'

});

gameEffectLayer.add(line);

var anim = new Kinetic.Animation(function(frame) {

var period = 2000;

if(frame.time>period)

{

anim.stop();

point.player = selectedChecker.player;

point.circle.setFill(selectedChecker.circle.attrs.fill);

selectedChecker.player = -1;

selectedChecker.circle.attrs.fill='';

selectedChecker = null;

gameLayer.clear().draw();

if(callBack)callBack();

}

}, gameEffectLayer);

anim.start();

gameEffectLayer.clear().draw();

}

function moveCheckerTo2(point)

{

gameEffectLayer.removeChildren();

var path = findPath(selectedChecker,point);

var points = getLinePointsArray(path);

points.shift();

points.shift();

var p = gridXyToXy(selectedChecker);

var c = new Kinetic.Circle({

x: p.x,

y: p.y,

radius: 15,

fill: userCheckerColors[selectedChecker.player],

stroke: 'black',

//strokeWidth: 1,

});

gameEffectLayer.add(c);

var anim = new Kinetic.Animation(function(frame) {

var period = 5000;

if(points.length<=1)

{

anim.stop();

point.player = selectedChecker.player;

point.circle.setFill(selectedChecker.circle.attrs.fill);

selectedChecker.player = -1;

selectedChecker.circle.attrs.fill='';

selectedChecker = null;

gameLayer.clear().draw();

gameEffectLayer.clear().draw();

if(callBack)callBack();

}

else

{

var cx = c.getX();

var cy = c.getY();

var x = points[0];

var y = points[1];

var m = (y-cy)/(x-cx);

var rate = frame /5000 \* (y>cy?1:-1);

var type = x>cx?"right":"left";

c.setX(cx+rate);

c.setY(cy+m\*rate);

if(type=="right" && cx>x || type=="left" && cx<=x)

{

x = points.shift();

y = points.shift();

}

}

}, gameEffectLayer);

gameEffectLayer.clear().draw();

anim.start();

}

function initGame(player)

{

var pics = ['./red.jpg','./yellow.jpg','green.jpg'];

var y = 20;

gameLayer.removeChildren();

for(var i=0;i<player.length;i++)

{

newPlayerZone(20,y,150,80,player[i].userName,pics[i],gameLayer);

gameLayer.add(newNowPlayerEffect(20-2,y-4,150+4,110+4,'blue',player[i].userId));

y+=130;

}

stage.find('.playerZoneEffect').each(function(a){a.hide()});

chessPoints = getInitChessPoint();

test();

}

function displayTurns(player)

{

showMessage(player.userName+"' turn.");

stage.find('.playerZoneEffect').each(function(a){a.hide()});

stage.find('#'+player.userId)[0].show();

gameLayer.clear().draw();

}

function showMessage(message)

{

var labelWidth = stage.getWidth();

var turnLabel = newLabel(-labelWidth,stage.getHeight()/2,message,labelWidth,50);

gameEffectLayer.add(turnLabel);

gameEffectLayer.clear().draw();

var anim = new Kinetic.Animation(function(frame) {

var period = 5000;

var scale = Math.cos(frame.time \* 2 \* Math.PI / period) \* 10+1;

var x = turnLabel.getX()+scale;

if(turnLabel.getX()+turnLabel.getWidth()<=10&&frame.time>period/2||frame.time>period/3\*4)

{

this.stop();

gameEffectLayer.removeChildren();

gameEffectLayer.clear();

}

turnLabel.setX(x);

}, gameEffectLayer);

anim.start();

}

function greyBackgroundEffect(layer,callBack,x)

{

var selectEffectLayer = new Kinetic.Rect({

x: x,

y: 0,

fill: 'black',

width: stage.getWidth(),

height: stage.getHeight(),

opacity :0.3

});

selectEffectLayer.on('click',function(){

layer.removeChildren();

layer.clear().draw();

if(callBack)callBack();

});

gameEffectLayer.add(selectEffectLayer);

}

function showCheckMessage(message,callBack)

{

var textField = newButton(0,stage.getHeight()/2,message,stage.getWidth(),30);

textField.on('click',function(){

gameEffectLayer.removeChildren();

gameEffectLayer.clear().draw();

if(callBack)callBack();

})

greyBackgroundEffect(gameEffectLayer,callBack);

gameEffectLayer.add(textField);

gameEffectLayer.clear().draw();

}

function isWin(playerNumber)

{

var defineWin = [];

//定義紅色領地0, TipPoint{2,2} move down & leftup

defineWin[0] = {TipPoint:{x:2,y:2}, dLeft:"down", dRight:"leftup"};

//defineWin[0] = {TipPoint:{x:-2,y:-2}, dLeft:"up", dRight:"rightdown"};

//定義黃色領地1, TipPoint{2,-4} move leftup & right

defineWin[1] = {TipPoint:{x:2,y:-4}, dLeft:"leftup", dRight:"right"};

//defineWin[1] = {TipPoint:{x:-2,y:4}, dLeft:"rightdown", dRight:"left"};

//定義綠色領地2, TipPoint{-4,-2} move right & down

defineWin[2] = {TipPoint:{x:-4,y:2}, dLeft:"right", dRight:"down"};

//defineWin[2] = {TipPoint:{x:4,y:-2}, dLeft:"left", dRight:"up"};

var check = defineWin[playerNumber];

return checkPlayer(check.TipPoint, check.dLeft, check.dRight, playerNumber);

function checkPlayer(TipPoint, leftDirection, rightDirection, playerNumber)

{

for(var i=0; i<3; i++)

{

for(var j=0; j<=i; j++)

{

var tempPoint = {x:TipPoint.x, y:TipPoint.y};

//move leftDirection

for(var k=0; k<i; k++)

{

tempPoint = getMovePoint(leftDirection, tempPoint);

}

//move rightDirection

for(var k=0; k<j; k++)

{

tempPoint = getMovePoint(rightDirection, tempPoint);

}

tempPoint = getPoint(tempPoint, chessPoints);

if(tempPoint.player != playerNumber)

return false;

}

}

return true;

}

}

script/chessBoard.js

var chessBoardCenter = {x:393.5,y:207};

var chessBoardGridEdge = 45;

var chessPoints;

var userCheckerColors = ["red","yellow","green"];

var selectedChecker;

function getInitChessPoint(number)

{

console.log("PlayerNum: " + number);

var point = {};

var moveDirection = getMoveDirection();

//定義紅色領地0 & player 0 所在之地, TipPoint{-2,-2} move up & rightdown

setPlayerAndDomain({x:-2,y:-2},"up","rightdown",0,0);

//定義綠色領地2, TipPoint{-4,-2} move right & down

setPlayerAndDomain({x:-4,y:2}, "right","down",-1,2);

//定義黃色領地1 & player 1 所在之地, TipPoint{-2,4} move rightdown & left

setPlayerAndDomain({x:-2,y:4}, "rightdown","left",1,1);

//定義紅色領地0, TipPoint{2,2} move down & leftup

setPlayerAndDomain({x:2,y:2}, "down","leftup",-1,0);

//定義綠色領地2 & player 2 所在之地, TipPoint{4,-2} move left & up

setPlayerAndDomain({x:4,y:-2}, "left","up",number==3?2:-1,2);

//定義黃色領地1, TipPoint{2,-4} move leftup & right

setPlayerAndDomain({x:2,y:-4}, "leftup","right",-1,1);

//定義中間無人區塊

var CenterPoint = {x:0,y:0};

for(var i=0; i<6; i++)

{

tempPoint = getMovePoint(moveDirection[i],CenterPoint);

tempPoint.player=-1;

tempPoint.domain=[-1];

point[tempPoint.x+","+tempPoint.y] = tempPoint;

}

CenterPoint.player=-1;

CenterPoint.domain=[-1];

point[CenterPoint.x+","+CenterPoint.y] = CenterPoint;

return point;

function setPlayerAndDomain(TipPoint, leftDirection, rightDirection, playerNumber, domain)

{

for(var i=0; i<3; i++)

{

for(var j=0; j<=i; j++)

{

var tempPoint = {x:TipPoint.x, y:TipPoint.y};

//move leftDirection

for(var k=0; k<i; k++)

{

tempPoint = getMovePoint(leftDirection, tempPoint);

}

//move rightDirection

for(var k=0; k<j; k++)

{

tempPoint = getMovePoint(rightDirection, tempPoint);

}

tempPoint = point[tempPoint.x+","+tempPoint.y] == undefined ? tempPoint : point[tempPoint.x+","+tempPoint.y];

tempPoint.player = tempPoint.player == undefined || tempPoint.player == -1 ? playerNumber : tempPoint.player;

//tempPoint.domain = tempPoint.domain == undefined ? {x:domain} : {x:tempPoint.domain.x,y:domain};

if(tempPoint.domain)

tempPoint.domain.push(domain);

else

tempPoint.domain = [domain];

//console.log(i + ", " + j + ": ");console.log(tempPoint);

point[tempPoint.x+","+tempPoint.y] = tempPoint;

}

}

}

}

function getPoint(point, points)

{

return points[point.x+","+point.y];

}

function gridXyToXy(point)

{

var x = (point.x+(point.y)/2) \*chessBoardGridEdge + chessBoardCenter.x;;

var y = -(point.y\*Math.sqrt(3)/2)\*chessBoardGridEdge + chessBoardCenter.y;

return {x:x,y:y};

}

function findAndRecordOnBoard(point)

{

var jumpStack = [];

var sureStack = [];

var moveDirection = getMoveDirection();

var tChessPoints = cloneChessPoint(chessPoints, ["x", "y", "player", "domain"]);

var selectedPoint = getPoint(point, chessPoints);

//console.log(tChessPoints);

var chessPoint;

for(var i=0; i<6; i++)

{// 對選到的點 往六個方向探詢 是否可以走

chessPoint = getPoint(getMovePoint(moveDirection[i], point), tChessPoints);

if(!chessPoint)

{

continue;

}

else if(chessPoint.player < 0)

{

sureStack.push(chessPoint);

}

}

jumpStack.push(getPoint(point, tChessPoints));

//console.log(jumpStack);

jump\_recursive(jumpStack, sureStack, moveDirection, tChessPoints);

//console.log(sureStack);

for(var i in sureStack)

{

tPoint = sureStack[i];

chessPoint = getPoint(tPoint, chessPoints);

if(isPointDomainBelongPlayer(chessPoint, selectedPoint.player))

{

chessPoint.computed = true;

}

}

}

function jump\_recursive(jumpStack, sureStack, moveDirection, tChessPoints)

{

if(jumpStack.length == 0)

{// 不用繼續遞迴

return;

}

//console.log("Start jump\_recursive");

var nextJumpStack = [];

var chessPoint;

for(var i in jumpStack)

{

tPoint = jumpStack[i];

for(var j in moveDirection)

{

var thisRoundMoveDirection = moveDirection[j];

if(tPoint[thisRoundMoveDirection] == true)

{// 此點此方向已經跳過

continue;

}

chessPoint = getPoint(getMovePoint(thisRoundMoveDirection, tPoint), tChessPoints);

if(!(chessPoint && chessPoint.player >= 0))

{// 確認此方向是否可以跳躍

continue;

}

chessPoint = getPoint(getMovePoint(thisRoundMoveDirection, tPoint, true), tChessPoints);

//console.log(chessPoint);

if(!chessPoint)

{

continue;

}

else if(chessPoint.player < 0)

{

tPoint[thisRoundMoveDirection] = true;

nextJumpStack.push(chessPoint);

sureStack.push(chessPoint);

}

}

}

//console.log("Next Round Jump");

//console.log(nextJumpStack);

jump\_recursive(nextJumpStack, sureStack, moveDirection, tChessPoints);

}

function findPath(startPoint,finalPoint)

{

var jumpStack = [];

var path = [];

var moveDirection = getMoveDirection();

var tChessPoints = cloneChessPoint(chessPoints, ["x", "y", "player", "domain"]);

var chessPoint;

startPoint = cloneOnePoint(startPoint);

finalPoint = cloneOnePoint(finalPoint);

for(var i=0; i<6; i++)

{

chessPoint = getPoint(getMovePoint(moveDirection[i], startPoint), tChessPoints);

if(!chessPoint)

{

continue;

}

else if(chessPoint.x==finalPoint.x&&chessPoint.y==finalPoint.y)

{

path.push(startPoint);

path.push(finalPoint);

return path;

}

chessPoint.mark = true;

}

return findPath\_recursive(startPoint,finalPoint,path,tChessPoints);

}

function findPath\_recursive(startPoint,finalPoint,path,tChessPoints)

{

var result = null;

var moveDirection = getMoveDirection();

path = clonePointsArray(path);

path.push(startPoint);

getPoint(startPoint,tChessPoints).mark = true;

for(var i=0; i<6; i++)

{

var middleChecker = getPoint(getMovePoint(moveDirection[i], startPoint),tChessPoints);

if(!middleChecker||middleChecker.player<0)continue;

var moveP = getMovePoint(moveDirection[i], startPoint,true)

chessPoint = getPoint(moveP, tChessPoints);

if(!chessPoint || chessPoint.mark || chessPoint.player >=0)

{

continue;

}

else if(chessPoint.x==finalPoint.x&&chessPoint.y==finalPoint.y)

{

path.push(finalPoint);

return path;

}

else

{

chessPoint.mark = true;

result = findPath\_recursive(chessPoint,finalPoint,path,tChessPoints);

if(result!=null)return result;

}

}

return result;

}

function isPointDomainBelongPlayer(point, playerNumber)

{

if(point.domain[0] == -1)

return true;

for(var i in point.domain)

if(point.domain[i] == playerNumber)

return true;

return false;

}

function getMovePoint(direction, point, isJump)

{

move = isJump == true ? 2 : 1;

switch(direction)

{

case 'up'://0

return {x:point.x,y:point.y+move};

break;

case 'down'://3

return {x:point.x,y:point.y-move};

break;

case 'left'://3

return {x:point.x-move,y:point.y};

break;

case 'right'://1

return {x:point.x+move,y:point.y};

break;

case 'leftup'://5

return {x:point.x-move,y:point.y+move};

break;

case 'rightdown'://2

return {x:point.x+move,y:point.y-move};

break;

}

}

function getMoveDirection()

{

return ["up", "right", "rightdown", "down", "left", "leftup"];

}

function clonePointsArray(array)

{

for(var i=0;i<array.length;i++)

{

delete array[i].circle;

}

return JSON.parse(JSON.stringify(array));

}

function cloneOnePoint(point)

{

var temp = point.circle;

point.circle='';

var obj = JSON.parse(JSON.stringify(point));

point.circle = temp;

return obj;

}

function cloneChessPoint(source, tag)

{

var destination = {};

for(var i in source)

{

destination[i] = {};

for (var property in tag)

{

destination[i][tag[property]] = deepCopy(source[i][tag[property]]);

}

}

return destination;

}

function deepCopy(obj)

{

if(obj == null || typeof(obj) !== 'object'){

return obj;

}

//make sure the returned object has the same prototype as the original

var ret = obj.constructor();

for(var key in obj)

{

ret[key] = deepCopy(obj[key]);

}

return ret;

}

script/ezWebChecker.js

var EzWebEvent = (function(){

function loginSuccessEvent()

{

turnToLobbyLayer();

EzWebGame.listRoomInfos();

}

function loginFailEvent(errorMsg)

{

alert(errorMsg);

}

function logoutEvent()

{

turnToLoginLayer();

}

function listRoomDoneEvent(roomInfos)

{

//console.log(JSON.stringify(roomInfos));

refreshLobbyRooms(roomInfos);

}

function createdRoomEvent(roomInfo)

{

turnToRoomLayer();

refreshRoomInfoLayer(roomInfo.Room);

refreshPlayersInRoomInfoLayer(roomInfo.Players);

}

function leavedRoomEvent()

{

turnToLobbyLayer();

EzWebGame.listRoomInfos();

}

function getRoomChangedEvent(roomInfo)

{

//refreshRoomInfoLayer(roomInfo.Room);

//console.log(roomInfo);

refreshPlayersInRoomInfoLayer(roomInfo.Players);

}

function roomJoinedEvent(roomInfo)

{

turnToRoomLayer();

refreshRoomInfoLayer(roomInfo.Room);

refreshPlayersInRoomInfoLayer(roomInfo.Players);

}

function roomStartdEvent(roomInfo)

{

turnToGameLayer();

initGame(roomInfo.Players);

}

function changeTurnEvent(player)

{

console.log(player.userName + "[" + player.userId + "]");

displayTurns(player);

}

function receiveStepEvent(instruction)

{

var step = JSON.parse(instruction.replace("\\\"","\""));

switch(step.Method)

{

case 'Select':

selectedChecker = getPoint(step.Point, chessPoints);

displaySelectCheckerEffect(selectedChecker);

displayPlaceToMove(selectedChecker);

break;

case 'MoveTo':

moveCheckerTo(getPoint(step.Point, chessPoints));

break;

case 'CancelSelect':

gameEffectLayer.removeChildren();

gameEffectLayer.clear().draw();

break;

}

}

function receiveCheckWinEvent(arriveId)

{

EzWebGame.replyCheck(isWin(EzWebGame.getUserTurnOrder(arriveId)));

}

function someoneFinishGameEvent(user)

{

showMessage(user.userName + " Finish Game");

}

function gameFinishEvent(rank)

{

console.debug(rank);

var messages = [];

var maxLength = 0;

for(var i=0;i<rank.length;i++)

{

var message = "#" + (i+1) + "\t\t" + rank[i].userName;

messages.push(message);

if(message.length > maxLength)

maxLength = message.length;

}

// 讓所有訊息等長

for(var i=0; i<messages.length;i++)

{

var message = messages[i];

for(var j=0;j<maxLength-message.length;j++)

{

messages[i] += " ";

}

}

showCheckMessage(messages.join("\n"), backToRoom);

function backToRoom()

{

turnToRoomLayer();

roomInfoLayer.show();

roomPlayerLayer.show();

}

}

return {

// 登入遊戲

onLoginSuccess: loginSuccessEvent,

onLoginFail: loginFailEvent,

onLogout: logoutEvent,

// 大廳中

onListRoomDone: listRoomDoneEvent,

onRoomCreated: createdRoomEvent,

// 房間中

onRoomLeaved: leavedRoomEvent,

onRoomChanged: getRoomChangedEvent,

onRoomJoined: roomJoinedEvent,

onRoomStarted: roomStartdEvent,

// 遊戲中

onChangeTrun: changeTurnEvent,

onReceiveStep: receiveStepEvent,

onCheckWin: receiveCheckWinEvent,

onAccomplishGame: someoneFinishGameEvent,

onGameFinish: gameFinishEvent

}

})();

function c\_createGameRoom()

{

var title = prompt('enter room title','Welcome');

var maxPlayer = prompt('enter maxPlayer','3');

if(isNaN(maxPlayer))maxPlayer=3;

EzWebGame.createGameRoom(title,2,maxPlayer);

}

script/ezWebGame.js

var EzWebGame = (function(){

var EzWebGameURL = "http://127.0.0.1/GameRound/";

var LocalLoginURL = "./login.php";

var TurnId = 0;

var gamePlayers = [];//遊戲開始的玩家

var request = (function(){

var queue = [];

var Key = '';

var LastKey = '';

var eventSSE;

function receiveKey(key)

{

if(queue.length > 0)

{

var node = queue.shift();

node.requestObject.url += key;

$.ajax(node.requestObject).done(node.doneRequest);

}

else

{

LastKey = key;

Key = key;

}

}

function send(requestObject, doneRequest)

{

if(Key != '')

{

requestObject.url += Key;

Key = '';

$.ajax(requestObject).done(doneRequest);

}

else

{

queue.push({"requestObject":requestObject, "doneRequest":doneRequest});

}

}

function clean()

{

Key = '';

LastKey = '';

}

function openSSE()

{

eventSSE = new EventSource(EzWebGameURL + 'Event/Request/' + LastKey);

console.log('openRequest()');

eventSSE.onmessage = function (event) {

console.debug(event.data);

events = JSON.parse(event.data).Events;

//console.log(new Date() + ": " + event.data);

for(var i=0; i<events.length ; i++)

{

switch(events[i]["Type"])

{

case 'RefreshRoomList':

EzWebEventCalls(EzWebEvent.onListRoomDone, events[i]["Param"]);

break;

case 'roomChanged':

EzWebEventCalls(EzWebEvent.onRoomChanged, events[i]["Param"]);

break;

case 'start':

gamePlayers = events[i]["Param"].Players;

EzWebEventCalls(EzWebEvent.onRoomStarted, events[i]["Param"]);

break;

case 'turn':

var param = JSON.parse(events[i]["Param"].replace("\\\"","\""));

TurnId = param.userId;

EzWebEventCalls(EzWebEvent.onChangeTrun, param);

break;

case 'message':

EzWebEventCalls(EzWebEvent.onReceiveStep, events[i]["Param"]);

break;

case 'checkWin':

var param = JSON.parse(events[i]["Param"].replace("\\\"","\""));

EzWebEventCalls(EzWebEvent.onCheckWin, param.WinnerId);

break;

case 'arrived':

console.debug(events[i]["Param"]);

if(events[i]["Param"]!="Cheat")

{

var param = JSON.parse(events[i]["Param"].replace("\\\"","\""));

EzWebEventCalls(EzWebEvent.onAccomplishGame, param);

}

break;

case 'rank':

var param = JSON.parse(events[i]["Param"].replace("\\\"","\""));

EzWebEventCalls(EzWebEvent.onGameFinish, param);

break;

default:

console.log(new Date() + "=> " + events[i]["Type"] + ':' + events[i]["Param"]);

}

}

};

eventSSE.onerror = function (event) {

console.log('eventSSE Error');

event.target.close();

openRequest();

}

}

function closeSSE()

{

eventSSE.close();

console.log('User Close Request');

}

function getId()

{

var infos = LastKey.split("\_");

return infos[1];

}

return {

receiveKey: receiveKey,

send: send,

clean: clean,

openSSE: openSSE,

closeSSE: closeSSE,

getUserId: getId

}

})();

function login()

{

$.ajax({

url: LocalLoginURL

}).done(function(data) {

console.log(data);

eval(data);

});

}

function logout()

{

var requestObject = {

url: EzWebGameURL + "user/logout/"

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

request.clean();

closeRequest()

EzWebEventCalls(EzWebEvent.onLogout);

}

}

function listRooms()

{

var requestObject = {

url: EzWebGameURL + "Room/ListRoomInfos/"

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

data = JSON.parse(data);

request.receiveKey(data.cKey);

EzWebEventCalls(EzWebEvent.onListRoomDone, data.Room);

}

}

function createRoom(title,minPlayer,maxPlayer)

{

var requestObject = {

url: EzWebGameURL + "Room/Create/" + title + "/" + minPlayer + "/" + maxPlayer + "/"

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

console.debug(data);

data = JSON.parse(data);

request.receiveKey(data.cKey);

TurnId = 0;

if(data.Wrong)alert(data.Wrong);

else

{

closeRequest();

openRequest();

var object = new Array();

EzWebEventCalls(EzWebEvent.onRoomCreated, {"Room":data.Room[0], "Players":data.Players});

}

}

}

function leaveRoom()

{

var requestObject = {

url: EzWebGameURL + "Room/Leave/"

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

console.debug(data);

data = JSON.parse(data);

request.receiveKey(data.cKey);

if(data.Wrong)alert(data.Wrong);

else

{

closeRequest();

openRequest();

EzWebEventCalls(EzWebEvent.onRoomLeaved);

}

}

}

function joinRoom(roomId)

{

var requestObject = {

url: EzWebGameURL + "Room/join/" +roomId+"/"

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

console.debug(data);

data = JSON.parse(data);

request.receiveKey(data.cKey);

TurnId = 0;

if(data.Wrong)alert(data.Wrong);

else

{

closeRequest();

openRequest();

EzWebEventCalls(EzWebEvent.onRoomJoined,{"Room":data.Room[0], "Players":data.Players});

}

}

}

function onReceiveFirstCKey(data)

{

if(data.Wrong!=null)

{

EzWebEventCalls(EzWebEvent.onLoginFail, data.Wrong)

}

else

{

request.receiveKey(data.cKey);

EzWebGame.openSSE();

EzWebEventCalls(EzWebEvent.onLoginSuccess)

}

}

function openRequest()

{

request.openSSE();

}

function closeRequest()

{

request.closeSSE();

}

function startRoom()

{

var requestObject = {

url: EzWebGameURL + "Exec/Start/"

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

console.debug(data);

data = JSON.parse(data);

request.receiveKey(data.cKey);

if(data.Wrong)alert(data.Wrong);

else

{

EzWebEventCalls(EzWebEvent.onRoomStarted, {"Players":data.Players});

gamePlayers = data.Players;

}

}

}

function getUserId()

{

return request.getUserId();

}

function getUserTurnOrder(userId)

{

userId = userId || request.getUserId();

for(var order in gamePlayers)

if(gamePlayers[order].userId == userId)

return order;

}

function getNowTurnUserOrder()

{

for(var order in gamePlayers)

if(gamePlayers[order].userId == TurnId)

return order;

}

function isTurnSelf()

{

return TurnId == getUserId();

}

function EzWebEventCalls(onEzWebEvent, data)

{

if(onEzWebEvent)

{

console.log(onEzWebEvent.name + ": " + JSON.stringify(data));

onEzWebEvent(data);

}

else

{

console.log("Not Found This Event: " + onEzWebEvent.toString());

}

}

function nextRound()

{

var requestObject = {

url: EzWebGameURL + "Exec/NextRound/"

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

console.debug(data);

data = JSON.parse(data);

request.receiveKey(data.cKey);

if(data.Wrong)alert(data.Wrong);

else

{

TurnId = data.NextRound.userId;

EzWebEventCalls(EzWebEvent.onChangeTrun, data.NextRound);

}

}

}

function sendMessage(instruction)

{

var requestObject = {

url: EzWebGameURL + "Exec/SendMessage/",

type: "POST",

data: {message: instruction}

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

console.debug(data);

data = JSON.parse(data);

request.receiveKey(data.cKey);

if(data.Wrong)alert(data.Wrong);

else

{

console.log("Send: " + instruction);

}

}

}

function arriveFinalStep()

{

var requestObject = {

url: EzWebGameURL + "Exec/ArriveFinalStep/",

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

console.debug(data);

data = JSON.parse(data);

request.receiveKey(data.cKey);

if(data.Wrong)alert(data.Wrong);

else

{

console.log("Inform the Player, I Won");

}

}

}

function replyCheck(isWin)

{

var requestObject = {

url: EzWebGameURL + "Exec/Reply/" + isWin + "/",

};

request.send(requestObject, doneRequest);

function doneRequest(data)

{

console.debug(data);

data = JSON.parse(data);

request.receiveKey(data.cKey);

if(data.Wrong)alert(data.Wrong);

else

{

console.log("Reply The Player is or isn't Win");

}

}

}

return {

// Prototype

cKey: onReceiveFirstCKey,

openSSE: openRequest,

isTurnSelf: isTurnSelf,

getNowTurnUserOrder:getNowTurnUserOrder,

getUserTurnOrder:getUserTurnOrder,

getUserId:getUserId,

// User

login: login,

logout: logout,

// Game

listRoomInfos: listRooms,

// Room

createGameRoom: createRoom,

leaveGameRoom: leaveRoom,

joinGameRoom: joinRoom,

startGameRoom: startRoom,

// Exec

doStep: sendMessage,

finishStep: nextRound,

finishGame: arriveFinalStep,

replyCheck: replyCheck

}

})();

script/uiComponent.js

function newLabel(px,py,Text,Width,labelFontSize)

{

if(!labelFontSize)labelFontSize=20;

var label = new Kinetic.Label({

x: px,

y: py,

opacity: 0.75,

});

label.add(new Kinetic.Tag({

fill: 'black',

lineJoin: 'round',

shadowColor: 'black',

shadowBlur: 0,

shadowOffset: 5,

shadowOpacity: 0.5,

width:Width,

}));

label.add(new Kinetic.Text({

text: Text,

fontSize: labelFontSize,

padding: 5,

fill: 'white',

align:'center',

width:Width,

}));

return label;

}

function newButton(px,py,buttonText,Width,labelFontSize)

{

if(!labelFontSize)labelFontSize=20;

var label = newLabel(px,py,buttonText,Width,labelFontSize);

label.on('mouseover',function(){

this.getTag().setFill('rgb(100,100,100)');

this.getLayer().draw();

})

label.on('mouseout',function(){

this.getTag().setFill('black');

this.getLayer().draw();

})

return label;

}

function newText(px,py,Text,FontSize,color)

{

if(!FontSize)FontSize=20;

if(!color)color='rgb(60,60,60)';

var simpleText = new Kinetic.Text({

x: px,

y: py,

text: Text,

fontSize: FontSize,

fill: color

});

return simpleText;

}

function newNowPlayerEffect(px,py,Width,Height,color,id)

{

return new Kinetic.Tag({

x:px,

y:py,

stroke: color,

strokeWidth: 5,

width:Width,

height:Height,

id:id,

name:'playerZoneEffect'

});

}

function newImage(X,Y,Width,Height,Src,layer,onLoadFunc)

{

var imageObj = new Image();

imageObj.onload = function() {

var image = new Kinetic.Image({

x: X,

y: Y,

image: imageObj,

width: Width,

height: Height

});

layer.add(image);

layer.draw();

if(onLoadFunc)onLoadFunc(image);

}

imageObj.src = Src;

}

function newPlayerZone(X,Y,Width,Height,PlayerName,Src,layer)

{

var shortSide = Width>Height?Height:Width;

var label = new Kinetic.Label({

x: X,

y: Y,

});

label.add(new Kinetic.Tag({

fill: 'white',

lineJoin: 'round',

shadowColor: 'black',

shadowBlur: 0,

shadowOffset: 5,

shadowOpacity: 0.5,

width:Width,

height:Height

}));

layer.add(label).add(newLabel(X,Y+shortSide,PlayerName,Width,20));

newImage(X+(Width-shortSide)/2,Y,shortSide,shortSide,Src,layer);

}

function newLobbyRoomZone(X,Y,Width,Height,RoomInfo,layer)

{

var labelFontSize = 20;

var group = new Kinetic.Group({

x: X,

y: Y,

opacity: 0.75,

});

var tag = new Kinetic.Tag({

fill: 'black',

lineJoin: 'round',

shadowColor: 'black',

shadowBlur: 0,

shadowOffset: 5,

shadowOpacity: 0.5,

width:Width-Height,

height:Height,

pointerDirection: 'right',

pointerWidth: Height/2,

pointerHeight: Height,

x:Height/2,

});

var triangle = new Kinetic.Shape({

drawFunc: function(context) {

context.beginPath();

context.moveTo(0, 0);

context.lineTo(Height/2, Height/2);

context.lineTo(0, Height);

context.lineTo(Height/2, Height);

context.lineTo(Height/2, 0);

context.closePath();

context.fillStrokeShape(this);

},

fill: 'black',

shadowColor: 'black',

shadowBlur: 0,

shadowOffset: 5,

shadowOpacity: 0.5,

});

group.add(tag).add(triangle);

group.add(new Kinetic.Text({

text: RoomInfo.title,

fontSize: labelFontSize,

fill: 'white',

align:'center',

width:Width,

y:(Height-labelFontSize)/2

}));

group.add(new Kinetic.Text({

text: RoomInfo.id,

fontSize: labelFontSize,

fill: 'white',

x:Height/2+15,

y:(Height-labelFontSize)/2

}));

group.add(new Kinetic.Text({

text: RoomInfo.now+'/'+RoomInfo.max,

fontSize: labelFontSize,

fill: 'white',

align:'left',

x:Width-Height/2-50,

y:(Height-labelFontSize)/2

}));

group.on('mouseover',function(){

tag.setFill('rgb(100,100,100)');

triangle.setFill('rgb(100,100,100)');

this.getLayer().draw();

})

group.on('mouseout',function(){

tag.setFill('black');

triangle.setFill('black');

this.getLayer().draw();

})

group.on('click',function(){EzWebGame.joinGameRoom(RoomInfo.id)});

layer.add(group);

}

script/uiLayer.js

var stage = new Kinetic.Stage({

container: 'container',

width: 600,

height: 410

});

var backgroundLayer = new Kinetic.Layer();

var loginLayer = new Kinetic.Layer();

var lobbyLayer = new Kinetic.Layer();

var roomLayer = new Kinetic.Layer();

var roomInfoLayer = new Kinetic.Layer();

var roomPlayerLayer = new Kinetic.Layer();

var gameLayer = new Kinetic.Layer();

var gameEffectLayer = new Kinetic.Layer();

var chessBoardLayer = new Kinetic.Layer();

var lobbyRoomsLayer;

//BcakgroundLayer

backgroundLayer.add(new Kinetic.Rect({

x: 0,

y: 0,

stroke: '#555',

strokeWidth: 5,

fill: '#ddd',

width: stage.getWidth(),

height: stage.getHeight(),

}));

//Login Layer

var loginLabel = newButton(200,stage.getHeight()/2+25,'login',200);

var registerLabel = newButton(200,stage.getHeight()/2+75,'register',200);

var ezWebCheckerLabel = newText(160,100,'EzWebChecker',50);

loginLabel.on('click',function(){EzWebGame.login()});

registerLabel.on('click',function(){window.open('http://127.0.0.1/GameRound/Member')})

loginLayer.add(loginLabel).add(ezWebCheckerLabel).add(registerLabel);

//LobbyLayer

var logoutLabel = newButton(0,100,'logout',80).on('click',function(){EzWebGame.logout();});

var createRoomLabel = newButton(0,150,'Create',80).on('click',function(){c\_createGameRoom();});

var refreshRoomListLabel = newButton(0,200,'refresh',80).on('click',function(){EzWebGame.listRoomInfos();});

lobbyLayer.add(newLabel(0,10,'Lobby',stage.getWidth()-10,45));

//roomInfoLayer.add(newLabel(stage.getWidth()-210,60,'MaxPlayer: '+room.max,200,20));

lobbyLayer.add(logoutLabel).add(refreshRoomListLabel).add(createRoomLabel);

//RoomLayer

var leaveRoomLabel = newButton(0,100,'Leave',80).on('click',function(){EzWebGame.leaveGameRoom();});

var startLabel = newButton(0,150,'Start',80).on('click',function(){EzWebGame.startGameRoom();});

roomLayer.add(leaveRoomLabel).add(startLabel);

//GameLayer

var text = newText(20,0,'');

var text2 = newText(320,50,'');

gameLayer.add(text).add(text2);

newImage(195,5,400,400,'./chess.jpg',chessBoardLayer,function(image){

image.on('mousemove', function(evt) {

//var mousePos = getMousePos(canvas, evt);

var message = 'Mouse pos: ' + evt.x + ',' + evt.y;

text.setText(message);

gameLayer.draw();

});

});

test();

function test()

{

for(var i in chessPoints)

{

var p = gridXyToXy(chessPoints[i]);

var c = new Kinetic.Circle({

x: p.x,

y: p.y,

radius: 15,

fill: userCheckerColors[chessPoints[i].player],

//stroke: chessPoints[i].player>=0?'black':'',

shadowColor:'black',

shadowOffset:3,

//strokeWidth: chessPoints[i].player>=0?2:''

});

c.attrs.point = chessPoints[i];

c.on('mousemove',function(event){

var point = event.targetNode.attrs.point;

var message = point.x+','+point.y;

text2.setText(message);

gameLayer.draw();

});

c.on('click',function(event){

clickChecker(event.targetNode.attrs.point);

})

chessPoints[i].circle = c;

gameLayer.add(c);

}

}

//Add layer to stage

stage.add(backgroundLayer).add(loginLayer).add(lobbyLayer).add(roomLayer).add(chessBoardLayer).add(gameLayer);

stage.add(roomInfoLayer).add(roomPlayerLayer).add(gameEffectLayer);

login.php

$gKey = "KlfQcRgxmNzzrjZRtH";

$EzWebGameURL = "http://127.0.0.1/GameRound/";

$getLKeyURL = $EzWebGameURL.'game/loadEzWebGameLib/'.$gKey;

$lKey = file\_get\_contents($getLKeyURL);

$loginURL = $EzWebGameURL.'member/login/'.$lKey;

echo "window.open('".$loginURL."', '', 'width=500, height=350')";