

# 소프트웨어프로젝트 I

기말프로젝트  
20181702 최혁태

# 앱 개발 순서

- API설계
- 게임 서버의 내부 설계
  - 핵심논리모듈(Game Logic) 개발
  - 게임 클래스 파일 개발
- WSGI 스크립트 작성
- 앱인벤터를 이용한 앱 개발

# 요구사항 명세(Requirement Specification)

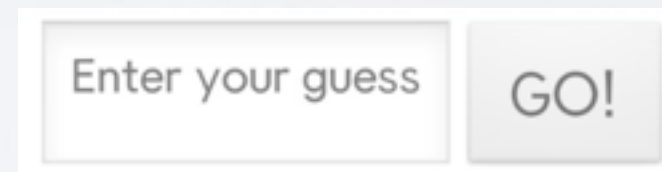
이 게임 시스템이 가져야 하는 요구사항

## 1. 기능적 요구사항

클라이언트-서버 구현  
웹서버(apache2)설정  
클라이언트 구현  
WSGI 스크립트  
예외처리

## 2. 비기능적 요구사항

마음속의 수에 대한 범위 지정  
게임 진행이 아닐 때는 GO 버튼 비활성화



Enter your guess

GO!

# 서버 API 설계

## 새로운 게임 시작

### 1. URL

<http://10.0.2.15/game/new>

### 2. Input: POST request body 를 이용

count = d

d: 사용자 입력으로, 랜덤으로 출력될 수의 범위를 지정한다.

### 3. Output (json 응답)

성공: {'code': 'success'}

실패: {'code': 'error', 'msg': 'count not given'}

# 서버 API 설계

## 숫자 맞추기

### 1. URL

<http://10.0.2.15/game/new>

### 2. Input: POST request body 를 이용

guess = d

d: 사용자 입력으로 숫자 맞추기를 시도한다.

### 3. Output (json 응답)

성공: {'code': 'success', 'compare': answer, 'trial': trial}

answer: 입력된 수와 정답을 비교해서 “Smaller”, “Greater”, “Success” 중 하나를 출력

trial: 시도한 횟수

실패: {'code': 'error', 'msg': 'wrong guess parameter'}

---

# API 구현 - WSGI

## WSGI 스크립트 작성

```
if environ['REQUEST_METHOD'] != 'POST':          #만약 포스트 요청이 아닐경우
    response = {'code': 'error', 'msg': 'wrong HTTP method'}  #에러메세지를 담은 dictionary
    error = True
```

```
if not error:
    try:                                           #environ['PATH_INFO']로부터 요청된 기능(API) 파악
        path = environ['PATH_INFO'].split('/')
        if len(path) == 2:
            method = path[1]
        else:
            response = {'code': 'error', 'msg': 'wrong API path'}
            error = True
    except:
        response = {'code': 'error', 'msg': 'wrong API path'}
        error = True
```

environ에 주어진 환경 변수로부터  
request path를 검사하고(길이체크)  
어떤 API가 호출되었는지를 판단한다

# API 구현 - WSGI

## WSGI 스크립트 작성

`if not error:`      #요청된 API에 따라 적절한 게임 드라이버 함수를 호출한다.

`if method == 'new':`

`response = new_game(d)`

`elif method == 'guess':`

`response = guess(d)`

`else:`

`response = {'code': 'error', 'msg': 'non-existent API method'}`

|  |
|--|
| URL의 맨 끝에 들어가는 내용에 따라<br>new 인 경우<br>guess 인 경우<br>아무것도 아닌경우로 나누어 처리 |
|--|

`status = '200 OK'`

`response_body = json.dumps(response)`      #json 형태를 가지는 HTTP response를 출력한다.

```
response_headers = [  
    ('Content-Type', 'application/json'),  
    ('Content-Length', str(len(response_body)))  
]
```

`start_response(status, response_headers)`

`return [response_body]`

---

# 게임의 핵심 로직 설계

게임의 구현에 있어 핵심이 되는 클래스를 정의한다.

## 1. 게임의 구현에 있어 핵심이 될 클래스

```
class Findnumber:
```

## 2. 이 클래스가 가져야 할 속성

number : 랜덤으로 지정되는 숫자

trial : 내가 시도한 횟수

---



# 게임의 핵심 로직 설계

## 3. 이 클래스가 가져야 할 메서드

```
def __init__(self):  
    self.number = 0  
    self.trial = 0
```

: 속성을 0으로 초기화한다.

```
def new_game(self, limit):  
    self.number = random.randint(1, limit)  
    self.trial = 0
```

: 새로운 게임을 시작하면 1부터 정해진 범위(limit)안에서 랜덤으로 숫자를 출력하고 시도횟수를 0으로 초기화한다.

```
def compare(self, num):  
    if num > self.number:  
        result = "Smaller"  
    elif num < self.number:  
        result = "Greater"  
    elif num == self.number:  
        result = "Success"  
    self.trial += 1  
  
    return result
```

: 내가 입력한 숫자(num)과 랜덤으로 정해진 숫자(self.number)의 크기를 비교해서 “Smaller”, “Greater”, “Success” 중 하나를 출력하고 한번 숫자를 입력 받을때마다 시도횟수(self.trial)를 하나씩 증가시킨다.

```
def gettrial(self):  
    return self.trial
```

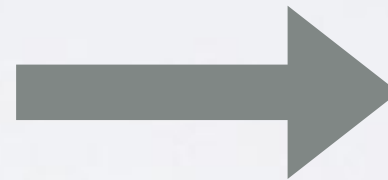
: 시도횟수(self.trial)을 반환한다(return)

---

# 게임의 핵심 로직 설계

게임을 임의로 테스트한다.

```
if __name__ == '__main__':  
    game = Findnumber()  
    game.new_game(100)  
  
    while(1):  
        mynumber = int(input("Your guess:"))  
        gameresult = game.compare(mynumber)  
        print(gameresult)  
        if gameresult == "Success":  
            print("SUCCESS in",game.gettrial(),"trials")  
            break
```



텍스트기반으로  
게임을 테스트할수 있는  
코드를 작성한다.

```
Your guess:50  
Greater  
Your guess:80  
Smaller  
Your guess:75  
Smaller  
Your guess:65  
Greater  
Your guess:70  
Smaller  
Your guess:67  
Greater  
Your guess:69  
Smaller  
Your guess:68  
Success  
SUCCESS in 8 trials
```

# WSGI 스크립트 및 추가 필요 모듈 구현

```
if __name__ == '__main__':  
    httpd = make_server(  
        'localhost',  
        8051,  
        application  
    )  
  
    httpd.serve_forever()
```

#localhost/8051이라는 서버를 만들고 유지한다.  
localhost의 터미널에서 curl을 이용하여  
텍스트기반으로 테스트한다.



```
choihyuktae@hyuktae:~/Desktop$ curl -d "count=100" http://localhost/game/new  
{"code": "success"}choihyuktae@hyuktae:~/Desktop$
```

# 서버통합

1. WSGI 스크립트를 적절한 위치에 배치한다.

`/var/www/game`

2. Apache 설정을 조정하여 URL과 스크립트를 연결한다

`/etc/apache2/conf-enabled/game.conf`로 설정파일 작성

3. 웹서버 재시작(변경된 설정이 적용되기위해)

`sudo service apache2 restart`

4. 터미널에서 curl을 이용하여 기본동작 테스트

웹 서버에 WSGI 스크립트가 올바르게 연동되는지 확인

---

# 서버통합

— **1** — 2 — 3 — 4

1. 세개의 파이썬 파일을 /var/www/game에 배치한다.

```
choihyuktae@hyuktae:~/Desktop$ sudo cp *.py /var/www/game
[sudo] password for choihyuktae:
choihyuktae@hyuktae:~/Desktop$ cd /var/www/game
choihyuktae@hyuktae:/var/www/game$ ls
WSGI.py  findnumber.py  game.py
```

# 서버통합

— | — **2** — 3 — 4

2. 새로운 파일을 /etc/apache2/conf-enabled 아래에 작성

`sudo vi /etc/apache2/conf-enabled/game.conf`

```
WSGIDaemonProcess game threads=1 home=/var/www/game
WSGIProcessGroup game
WSGIPythonPath /var/www/game
WSGIScriptAlias /game /var/www/game/WSGI.py
~
~
~
~
~
~
"game.conf" [readonly] 4 lines, 147 characters
```

game.py와 findnumber.py 내의 객체들을 import 할 수 있게한다.

# 서버통합

— | — 2 — **3** — 4

## 3. 웹서버 재시작(변경된 설정이 적용되기위해)

`sudo service apache2 restart`

```
choihyuktae@hyuktae:/var/log/apache2$ sudo service apache2 restart
{"code": "success"}choihyuktae@hyuktae:/var/www/game$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:49:69:c8
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::bdcc:59b:7369:1e2b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:269362 errors:0 dropped:0 overruns:0 frame:0
          TX packets:111871 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:244161007 (244.1 MB)  TX bytes:6881988 (6.8 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:650 errors:0 dropped:0 overruns:0 frame:0
          TX packets:650 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:59248 (59.2 KB)  TX bytes:59248 (59.2 KB)
```



# 서버통합

— | — 2 — 3 — 4

## 4.터미널에서 curl을 이용하여 기본동작 테스트

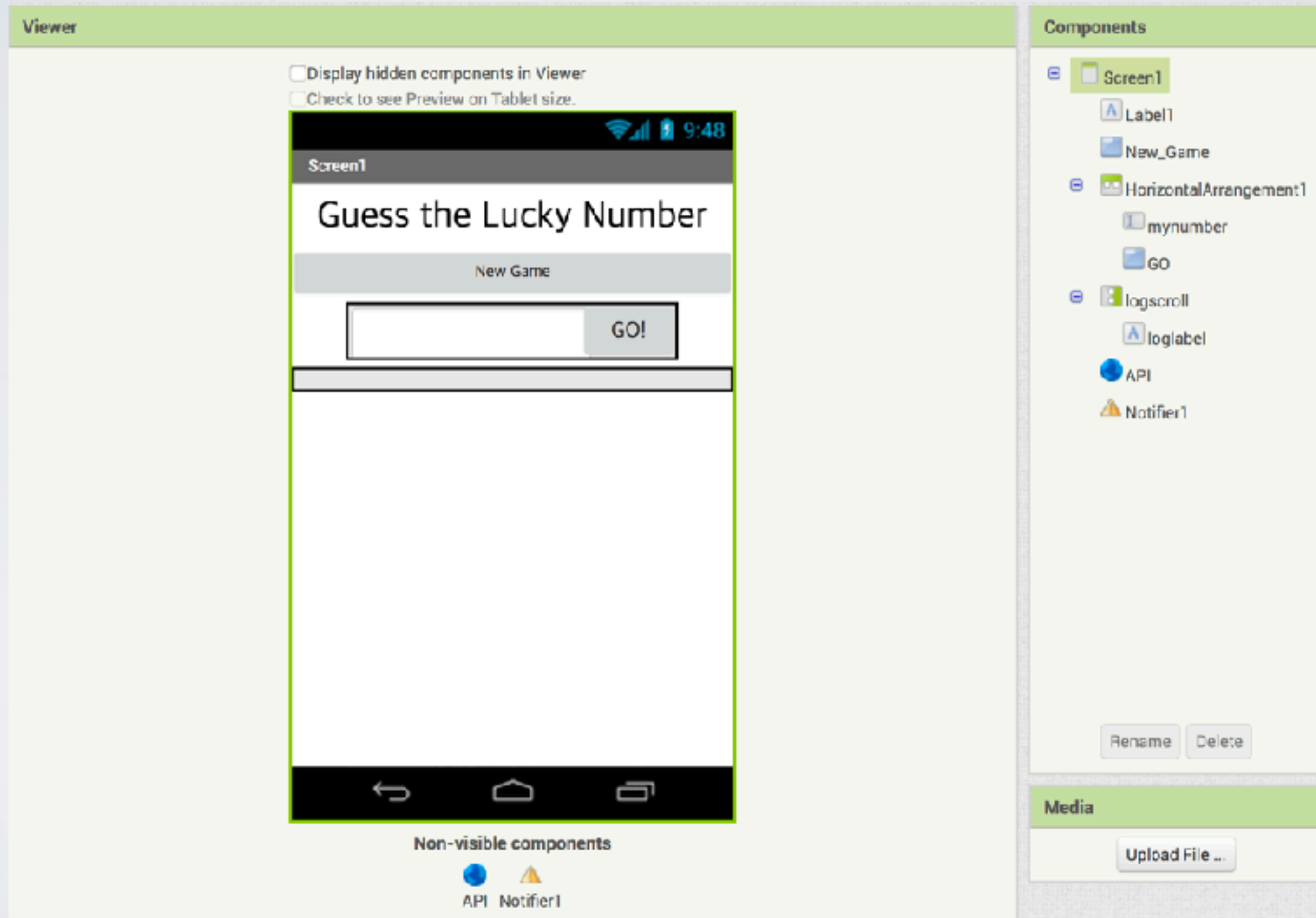
웹 서버에 WSGI 스크립트가 올바르게 연동되는지 확인

```
choihyuktae@hyuktae:/var/www/game$ curl -d "count=10" http://localhost/game/new  
{"code": "success"}choihyuktae@hyuktae:/var/www/game$
```



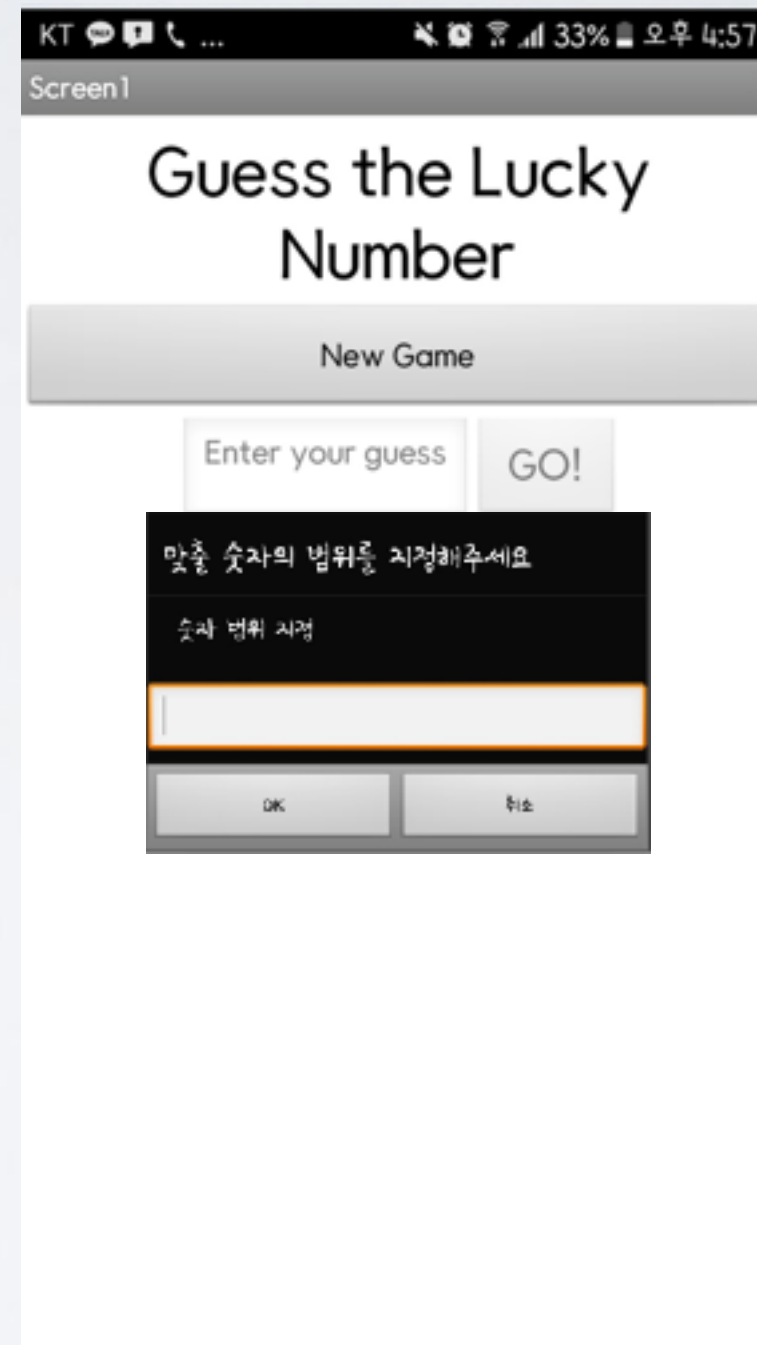
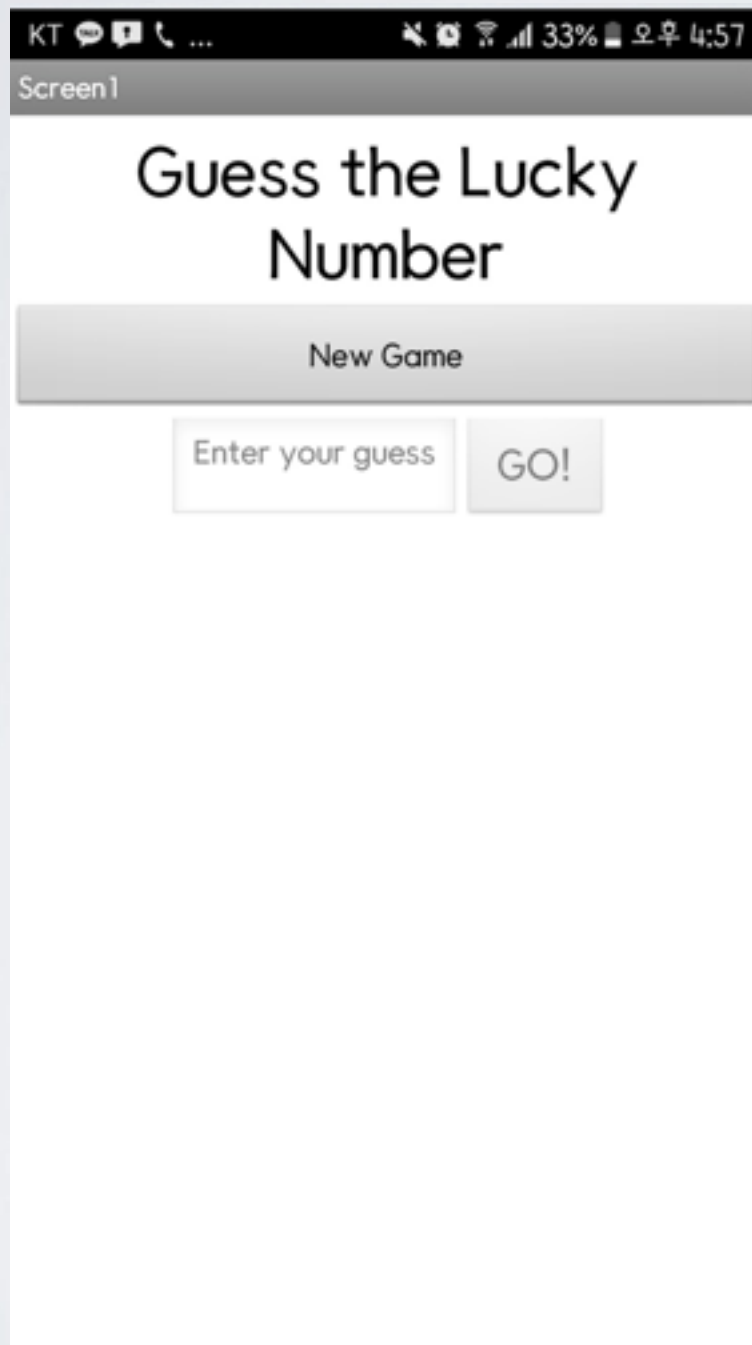
# 클라이언트 개발

## UI를 설계한다.



# 클라이언트 개발

## UI를 설계한다.



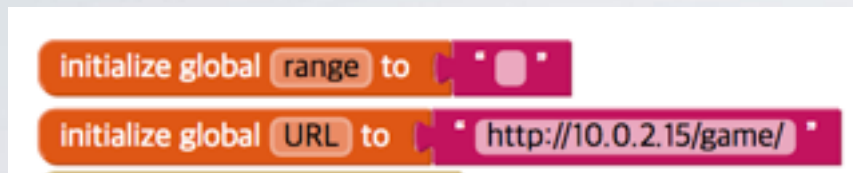
# 클라이언트 개발

## UI를 설계한다.



# 클라이언트 개발

## 블럭코딩



```
initialize global range to ""
initialize global URL to "http://10.0.2.15/game/"
```

: 변수 설정



```
when Screen1.Initialize
do set GO.Enabled to false
```

: 실행중이지 않을때는  
GO버튼 비활성화

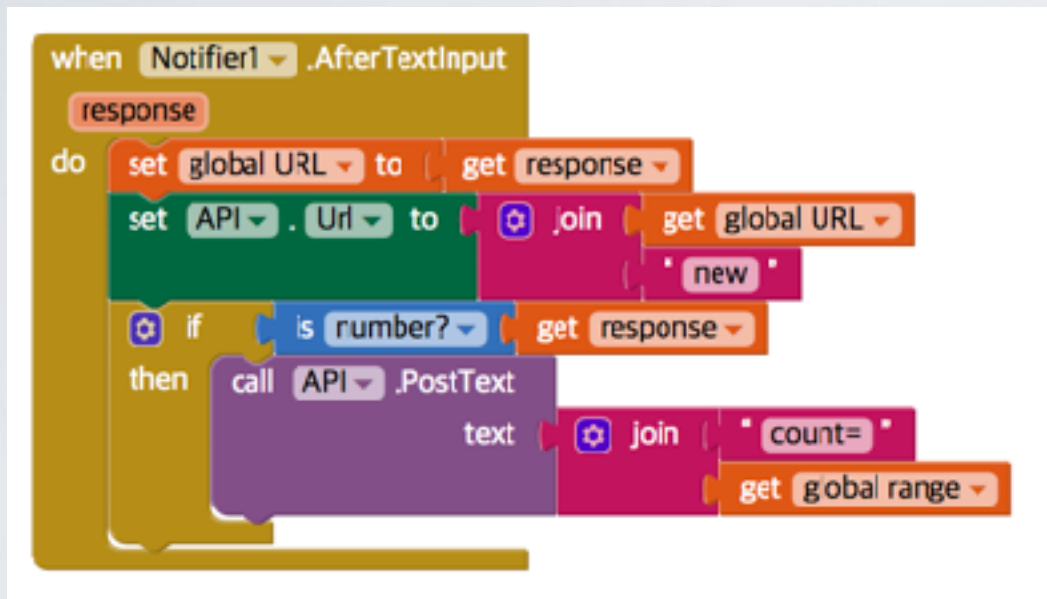


```
when New_Game.Click
do set GO.Enabled to false
  call Notifier1.ShowDialog
    message "숫자 범위지정"
    title "맞출 숫자의 범위를 지정해주세요"
    cancelable true
```

: New Game을 누르면  
나오는 설명 입력

# 클라이언트 개발

## 블럭코딩



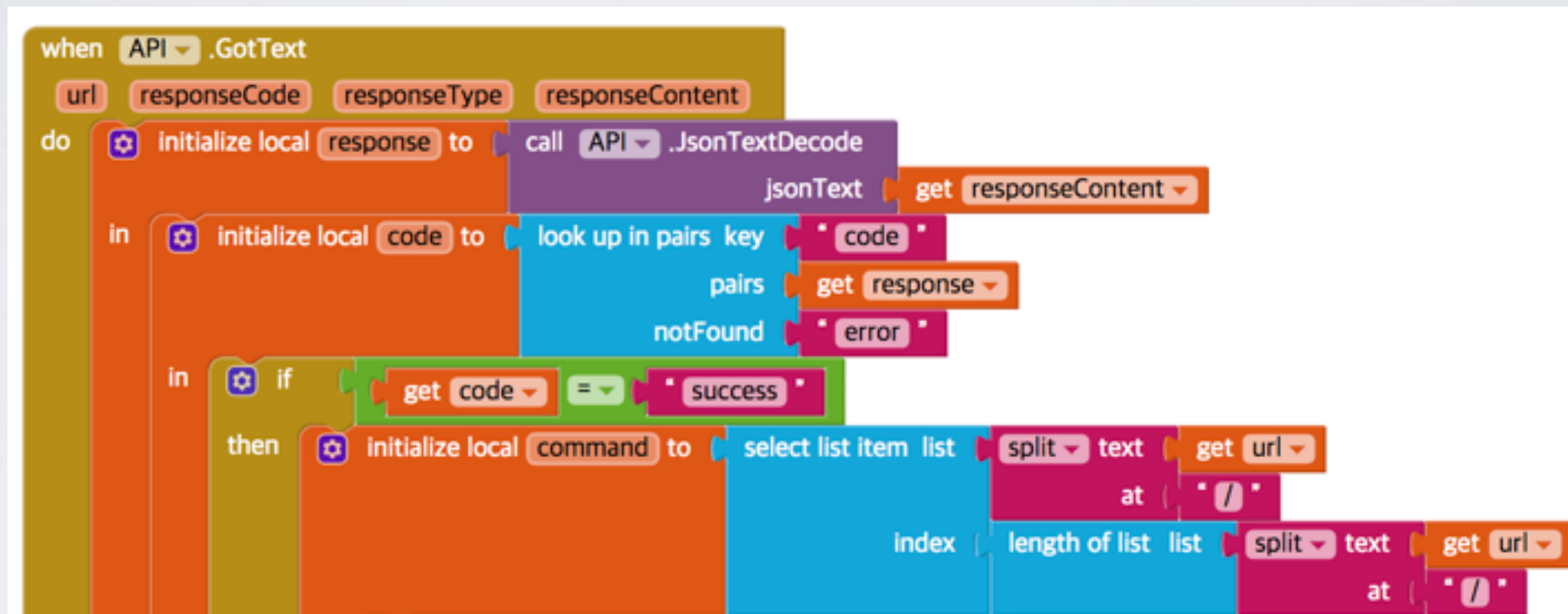
: 범위를 입력받으면 API를 통해 count에 들어간다.



: GO버튼을 클릭하면 설정된 URL로 연결하고 입력한 숫자가 guess에 들어간다.

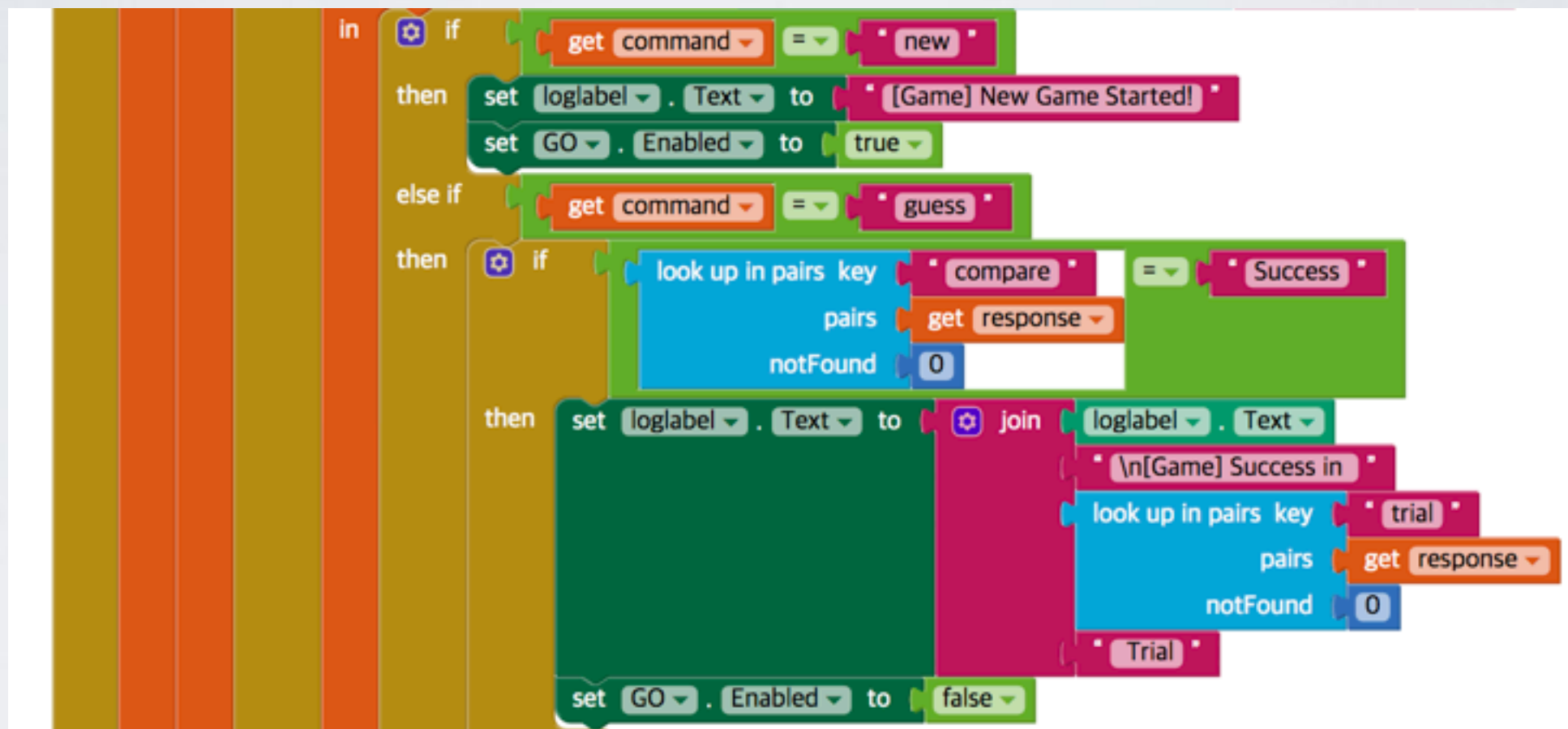
# 클라이언트 개발

## 블럭코딩



# 클라이언트 개발

## 블럭코딩



# 클라이언트 개발

## 블럭코딩

