클라이언트-서버 프로젝트

소프트웨어학과 20170126 정민선

목차

- 요구사항 명세
- 클라이언트 UI 설계

요구사항 명세

기능적 요구사항

- 게임 시작 버튼을 누르면 사용자가 최대 숫자를 설정할 수 있어야한다.
- 1부터 사용자가 설정한 숫자 범위 사이에서 랜덤으로 목표 숫자가 설정되어야 한다.
- 앱이 처음 실행되었을 때는 맞추기 버튼이 비활성화 되었다가, 게임이 시작되면 활성화되고, 게임이 종료되면 다시 비활성화 되어야한다.

비기능적 요구사항

- 클라이언트의 경우 App Inventor로 UI로 구현하고, POST방식을 이용하여 서버로 데이터를 보낸다.
- 서버의 경우 Apache2 웹서버와 WSGI를 사용하고 json 형태의 문자열을 클라이언트로 전달한다.
- 예외가 발생한 경우, 어떤 예외인지 표시한다.

게임 로직 설계

NumberGame.py

클래스 - NumberGame

```
class NumberGame:

    def newGame(self, count):
        pass

    def guess(self, userGuess):
        pass

    def getGuessCount(self):
        pass
```

- NumberGame이라는 클래스에 세개의 메소드 생성.

생성자

```
class NumberGame:
    def __init__(self):
        self.digits = 0
        self.trials = 0
```

- 생성자에서 랜덤 수 (마음 속 수)와 시도 횟수를 담을 변수를 초기화 시킨다.

메소드 – newGame (게임 시작)

```
import random
```

```
def newGame(self, count):
    self.digits = random.randint(1, count)
    self.trials = 0
```

- 사용자가 최대값을 설정할 수 있게 하기 위해 count를 매개변수로 설정하여 입력 받는다.
- 1부터 count까지의 범위사이의 random 수를 하나 추출하여 digits라는 변수에 저장
- 시도 횟수를 담을 trials변수는 0으로 초기화

메소드 – guess (숫자 판별)

```
def guess(self, userGuess):
    self.trials += 1

    if userGuess < self.digits:
        msg = "Greater"
    elif userGuess > self.digits:
        msg = "Smaller"
    else:
        msg = "Success"

    return msg
```

- 사용자로부터 숫자를 맞추도록 하기 위해 userGuess라는 매개변수를 설정하여 입력 받는다.
- 함수가 실행될 때 마다 시도횟수를 1씩 증가 시킨다.
- 사용자가 입력한 수(userGuess)가 랜덤(마음 속) 수(digits)와 비교하여 메시지를 출력 한다.
- 이 함수는 두 수를 비교한 뒤 메시지(msg)를 리턴 한다.

메소드 – getGuessCount (시도 횟수)

```
def getGuessCount(self):
    return self.trials
```

- trials 변수에 저장된 시도횟수를 리턴 한다.

게임 드라이버

game.py

게임 드라이버

```
from NumberGame import NumberGame

game = NumberGame()

def new_game(d):
    pass

def guess(d):
    pass
```

- NumberGame.py 파일로부터 클래스 NumberGame을 불러온다.
- 클래스 안의 메소드들을 사용하기 위해 game이라는 변수를 사용하여 객체를 생성한다.
- 메소드 두개를 구현한다. (뒤에서 자세한 설명 참고)

메소드 – new_game

```
def new_game(d):
    try:
        count = int(d.get('count',[''])[0])
    except:
        return{'code':'error', 'msg':'count not given'}

    game.newGame(count)

    return {'code':'success'}
```

- count값이 제대로 전달되지 않으면 except문이 실행된다.
- count값이 제대로 전달이 되면 NumberGame 클래스 안에 있는 newGame 메소드에 인자로 전달되고, {'code' : 'success'}를 리턴한다.

메소드 – guess

```
def guess(d):
    try:
        guess = int(d.get('guess', [''])[0])
    except:
        return {'code':'error', 'msg':'wrong guess parameter'}

    userGuess = game.guess(guess)
    trials = game.getGuessCount()

    return {'code':'success', 'msg': userGuess, 'trials':trials}
```

- guess값이 제대로 전달되지 않으면 except문이 실행된다.
- guess값이 제대로 전달되면, NumberGame 클래스 안의 guess메소드로 인자를 전달하여 받은 리턴값을 userGuess라는 변수에 저장하고, getGuessCount를 호출하여 얻은 시도횟수를 trials라는 변수에 저장한다. 그리고 마지막 줄을 리턴한다.

WSGI 스크립트

wsgi.py

```python
from cgi import parse_qs
import json
from game import new_game, guess


def application(environ, start_response):

    error = False

    if environ['REQUEST_METHOD'] != 'POST':
        response = {'code': 'error', 'msg': 'wrong HTTP method'}
        error = True

    if not error:
        try:
            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

    try:
        request_body_size = int(environ.get('CONTENT_LENGTH', '0'))
    except ValueError:
        request_body_size = 0
    request_body = environ['wsgi.input'].read(request_body_size)
    d = parse_qs(request_body)

    if not error:
        if method == 'new':
            response = new_game(d)
        elif method == 'guess':
            response = guess(d)
        else:
            response = {'code': 'error', 'msg': 'non-existent API method'}
    status = '200 OK'
    response_body = json.dumps(response)

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

    start_response(status, response_headers)
    return [response_body]
```

- POST값에 대한 예외처리.
- 주소에 대한 예외처리
- 주소 맨 뒤 값이 new인지 guess인지 아무것도 없는지에 따라 다르게 실행.
- 각 상황에 맞는 딕셔너리를 json 라이브러리를 이용하여 문자열로 덤프
- json형식으로 리턴

서버 설정 파일

/etc/apache2/conf-enabled/game.conf

서버 설정 파일

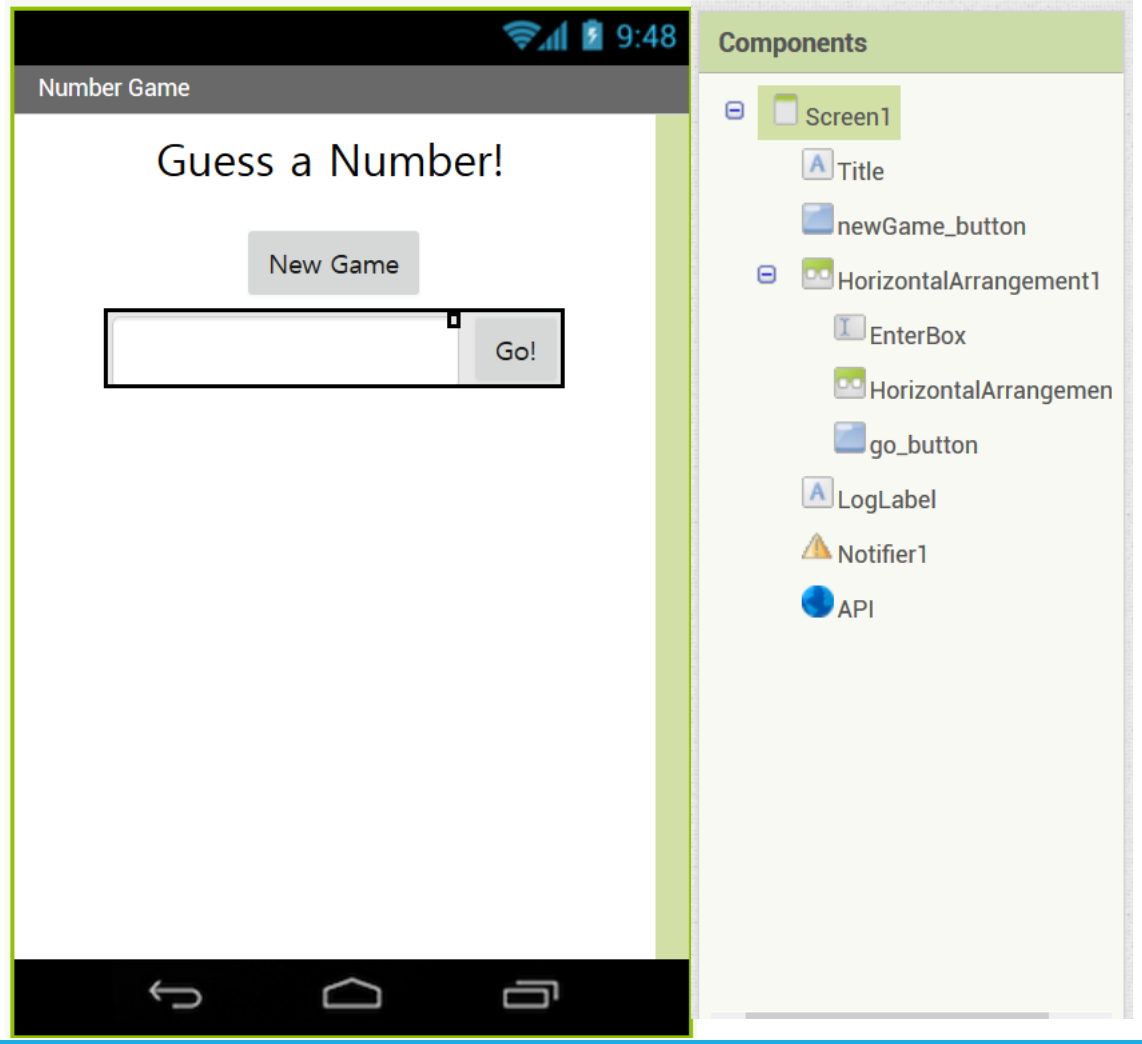
```
WSGIDaemonProcess game threads=1 home=/var/www/game
WSGIProcessGroup game
WSGIPythonPath /var/www/game
WSGIScriptAlias /game /var/www/game/wsgi.py
```

- 프로세스가 항상 실행되어야 하므로 daemon으로 설정한다.
- 충돌을 막기 위해 thread는 1개만 설정한다.
- NumberGame.py와 game.py 내의 객체들을 import 할 수 있도록 path를 설정한다.
- 주소 끝에 /game이 있으면 /var/www/game/wsgi.py를 실행한다.

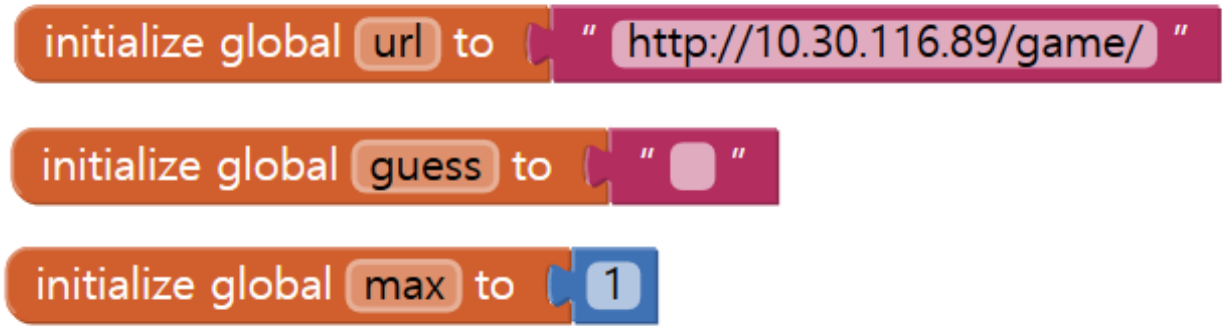
게임 클라이언트

App Inventor

Designer

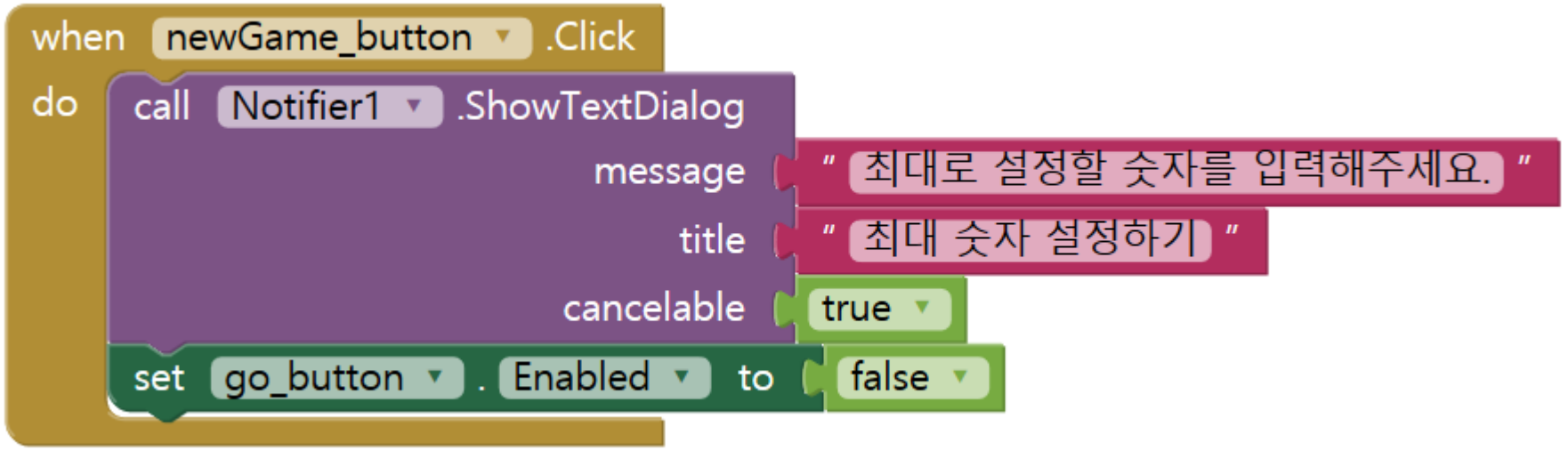

9:48
Number Game
Guess a Number!
New Game
Go!
Components
Screen1
Title
newGame_button
HorizontalArrangement1
EnterBox
HorizontalArrangemen
go_button
LogLabel
Notifier1
API

변수 설정, 초기화



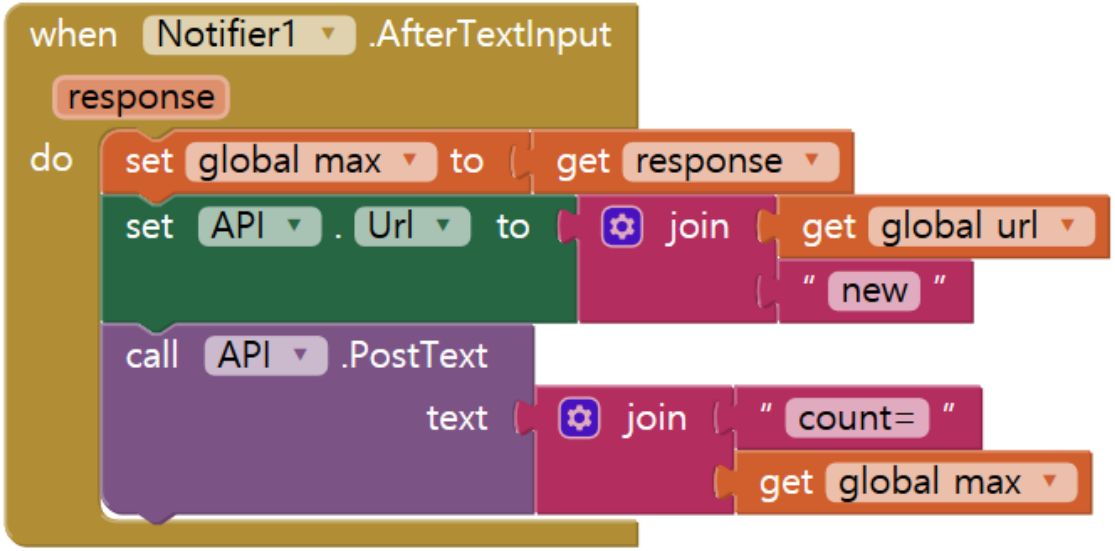
- url 주소와 사용자가 맞추기 위해 입력할 값, 사용자가 정하는 최댓값을 저장할 변수를 만들고 초기화한다.

newGame 버튼을 눌렀을 때



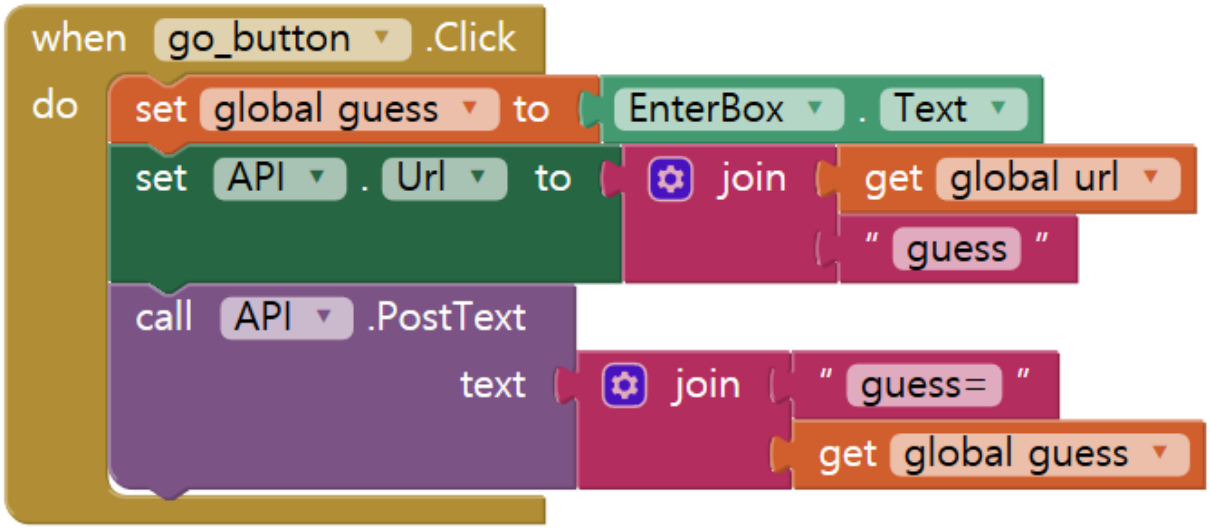
- newGame 버튼을 누르면 최댓값을 설정할 수 있게끔 알림창이 뜨게 하고, go 버튼은 아직 비활성화시킨다.

알림창에 숫자를 입력한 뒤



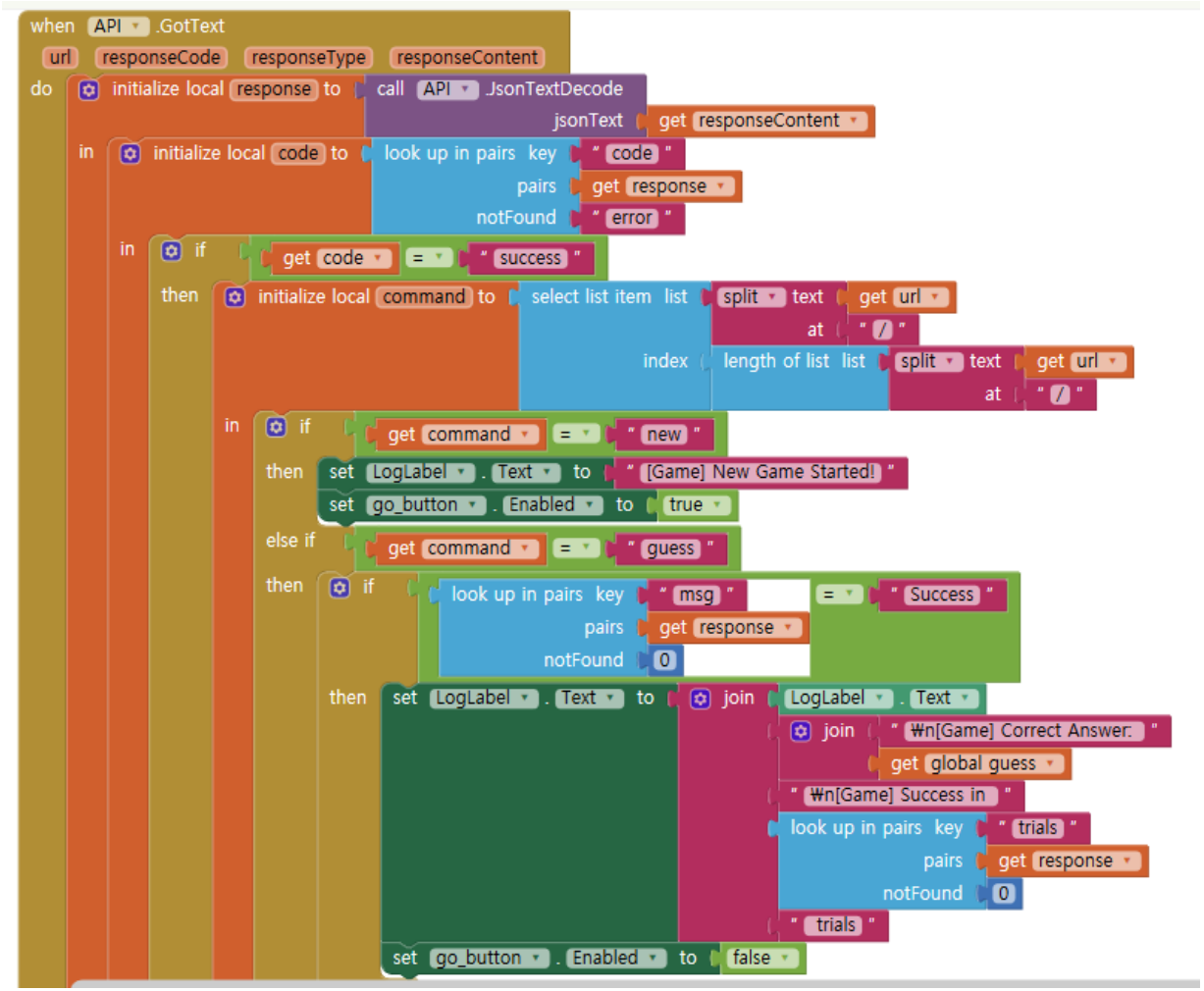
- 알림창에 적힌 숫자를 최댓값으로 설정한다.
- 기본 주소에 new를 추가한 뒤, “count=max” 형태로 최댓값을 전송한다.

Go 버튼을 눌렀을 때



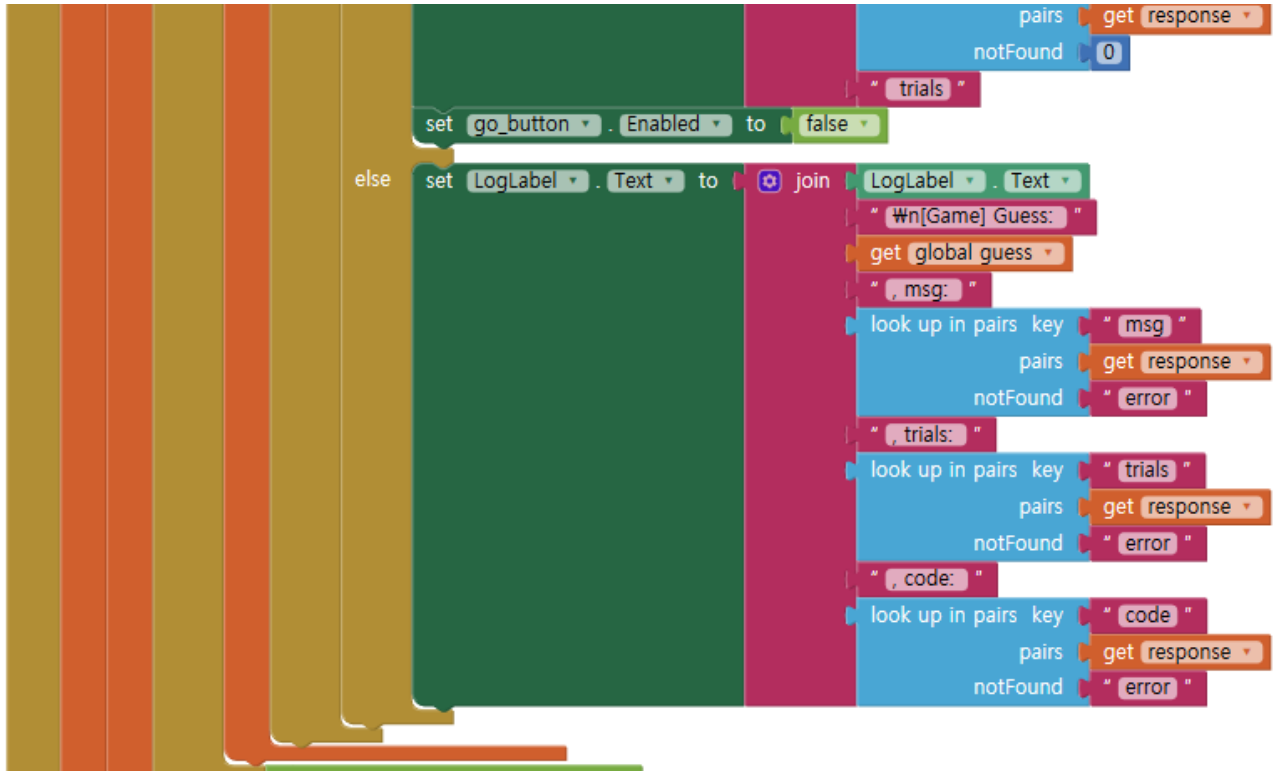
- Go 버튼 옆의 텍스트 창에 입력한 숫자를 변수 guess에 저장한다.
- 기본 주소에 guess를 추가한 뒤 “guess=guess”형태로 사용자의 입력 값을 전송한다.

API가 text를 받았을 때 (code가 success일 때)



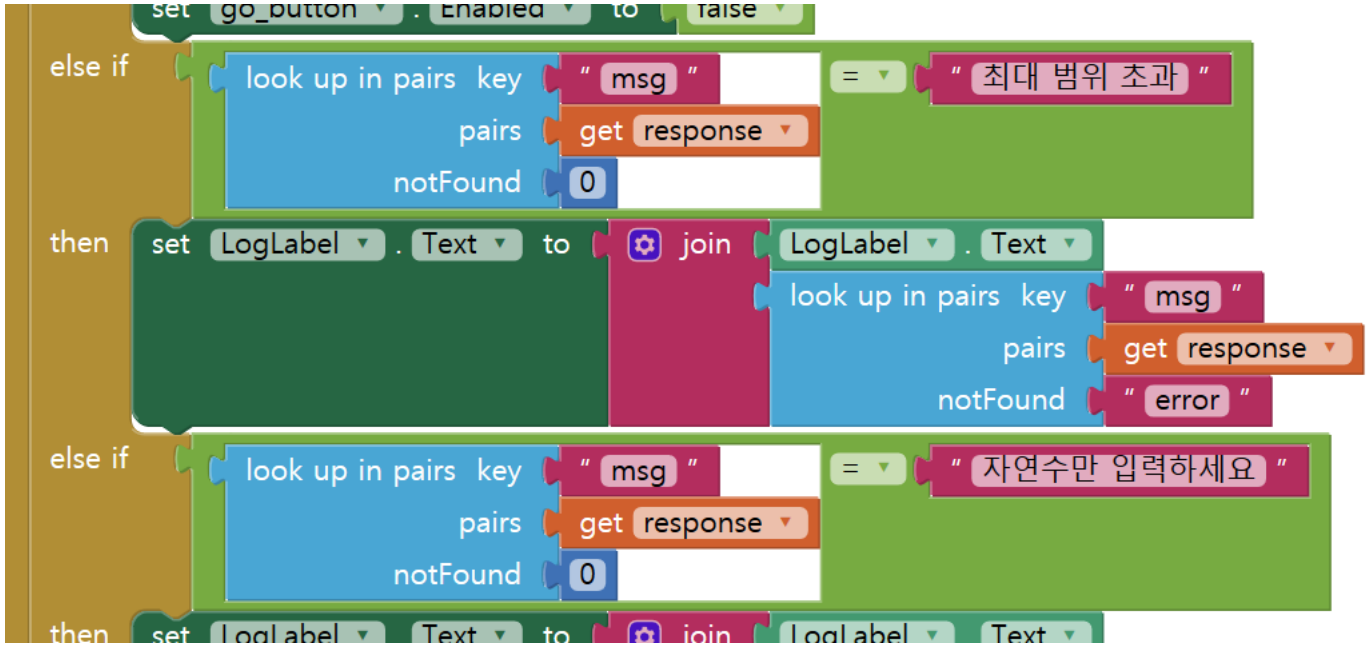
- new를 주소로 받으면 게임을 시작한다는 메시지를 출력한 뒤, go 버튼을 활성화시킨다.
- guess를 주소로 받고, 사용자가 랜덤 값을 맞추면 답과 시도 횟수를 출력한다.

API가 text를 받았을 때 (code가 success일 때)



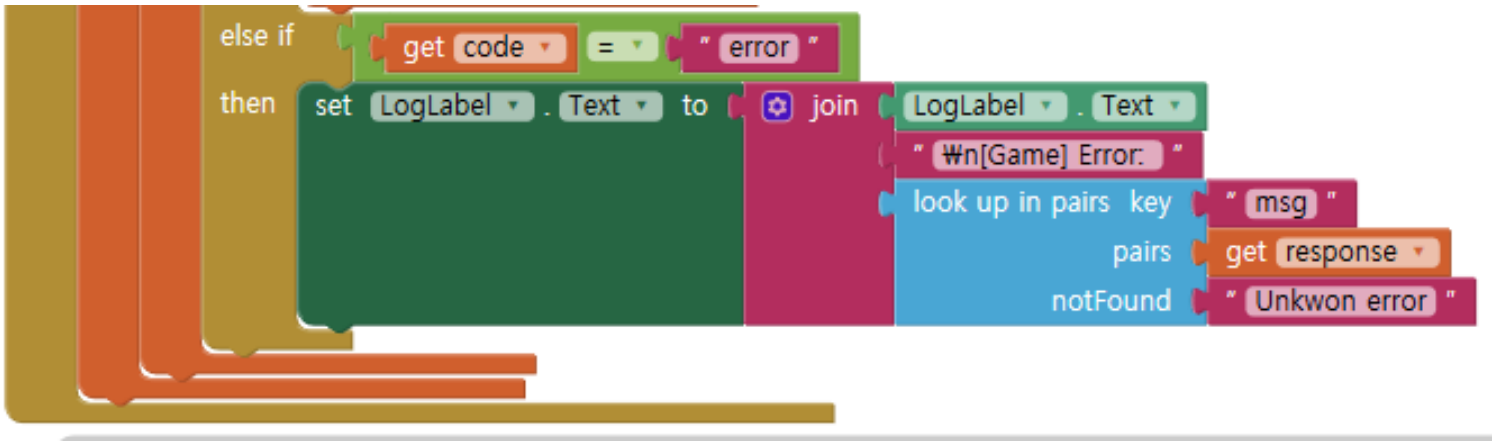
- guess를 주소로 받고, 사용자가 랜덤 값을 못 맞추면 greater나 small, 시도 횟수, 서버 코드를 출력한다.

API가 text를 받았을 때 (code가 success일 때)



- 최대값으로 설정한 값보다 큰 값을 입력한 것에 대한 예외처리
- 자연수가 아닌 수를 입력한 것에 대한 예외처리

API가 text를 받았을 때 (code가 error일 때)



- 에러 메시지를 출력한다.

테스트

Text 테스트

```
lucy@lucy-900X3M:~$ cd /var/www/game
lucy@lucy-900X3M:/var/www/game$ curl -d "count=100" http://192.168.0.24/game/new
{"code": "success"}lucy@lucy-900X3M:/var/www/game$
lucy@lucy-900X3M:/var/www/game$ curl -d "guess=20" http://192.168.0.24/game/guess
{"msg": "Greater", "trials": 1, "code": "success"}lucy@lucy-900X3M:/var/www/game$
lucy@lucy-900X3M:/var/www/game$ curl -d "guess=30" http://192.168.0.24/game/guess
{"msg": "Greater", "trials": 2, "code": "success"}lucy@lucy-900X3M:/var/www/game$
lucy@lucy-900X3M:/var/www/game$ curl -d "guess=40" http://192.168.0.24/game/guess
{"msg": "Smaller", "trials": 3, "code": "success"}lucy@lucy-900X3M:/var/www/game$
lucy@lucy-900X3M:/var/www/game$ curl -d "guess=35" http://192.168.0.24/game/guess
{"msg": "Smaller", "trials": 4, "code": "success"}lucy@lucy-900X3M:/var/www/game$
lucy@lucy-900X3M:/var/www/game$ curl -d "guess=34" http://192.168.0.24/game/guess
{"msg": "Smaller", "trials": 5, "code": "success"}lucy@lucy-900X3M:/var/www/game$
lucy@lucy-900X3M:/var/www/game$ curl -d "guess=33" http://192.168.0.24/game/guess
{"msg": "Smaller", "trials": 6, "code": "success"}lucy@lucy-900X3M:/var/www/game$
lucy@lucy-900X3M:/var/www/game$ curl -d "guess=32" http://192.168.0.24/game/guess
{"msg": "Success", "trials": 7, "code": "success"}lucy@lucy-900X3M:/var/www/game$
```

App Inventor 테스트