1. 음성인식 서비스 (네이버 AI 기술 이용) - 지현

음성인식 버튼을 통하여 앱 기능을 이용할 수 있습니다. 다음 병원 예약이나,

    복용알림을 알려주고, 약을 검색해주고, 약 추천도 받습니다.

\* API 사용 전 준비 사항

1. 콘솔의 **AI·Application Service > AI·NAVER API > Application**에서 애플리케이션을 등록합니다.(자세한 방법은 [Application 사용 가이드](https://docs.ncloud.com/ko/naveropenapi_v3/application.html) 참고)
2. **AI·Application Service > AI·NAVER API > Application**에서 등록한 애플리케이션을 선택해 Client ID와 Client Secret값을 확인합니다.
3. **AI·Application Service > AI·NAVER API > Application**의 **변경** 화면에서 **Clova Speech Recognition**가 선택되어 있는지 확인합니다. 선택되어 있지 않으면 429 (Quota Exceed)가 발생하니 주의하시기 바랍니다.

\* Android 기준

**1. 음성 인식 버튼 구현**

**2. 음성 인식 관련 API – 약을 검색해준다.**

**2-1. CSR**

Clova Speech Recognition API(이하 CSR API)는 사용자의 음성 입력을 스트리밍 형태로 입력받은 후 음성 인식한 결과를 텍스트로 반환하는 API

**2-2. CSR Android API 사용하기**

- 1. 다음 구문을 app/build.gradle 파일에 추가합니다.

repositories {

jcenter()

}

dependencies {

compile 'com.naver.speech.clientapi:naverspeech-ncp-sdk-android:1.1.6'

}

- 2. 다음과 같이 Android Manifest 파일(AndroidManifest.xml)을 설정합니다.

● 패키지 이름 : manifest 요소의 package 속성 값이 [사전 준비사항](https://docs.ncloud.com/ko/naveropenapi_v3/speech/recognition-sdk.html#Preparation)에서 등록한 **안드로이드 앱 패키지 이름**과 같아야 합니다.

● 권한 설정 : 사용자의 음성 입력을 마이크를 통해 녹음해야 하고 녹음된 데이터를 서버로 전송해야 합니다. 따라서, android.permission.INTERNET와 android.permission.RECORD\_AUDIO에 대한 권한이 반드시 필요합니다.

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.naver.naverspeech.client"

android:versionCode="1" android:versionName="1.0" >

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" /><uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

- 3. (선택) proguard-rules.pro 파일에 다음을 추가합니다. 아래 코드는 앱을 보다 가볍

고 안전하게 만들어줍니다.

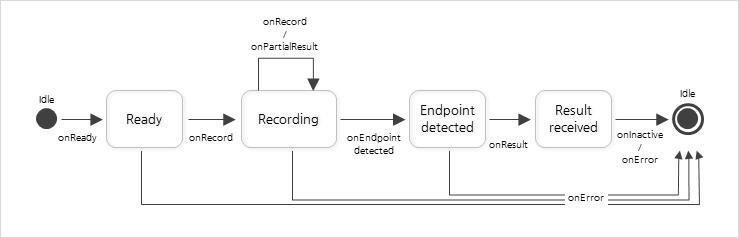
-keep class com.naver.speech.clientapi.SpeechRecognizer {

protected private \*;

}

- **Note!**

네이버 Open API는 Android SDK 버전 10 이상을 지원합니다. 따라서, *build.gradle* 파일의 minSdkVersion 값을 이에 맞게 설정해야 합니다.

클라이언트는 "준비", "녹음", "중간결과 출력", "끝점 추출", "최종결과 출력"과 같은 일련의 이벤트 흐름을 수행합니다. 애플리케이션 개발자는 SpeechRecognitioinListener 인터페이스를 상속받아 해당 이벤트가 발생할 때 처리할 동작을 구현하면 됩니다. 

- **Note!**

API에 대한 자세한 설명은 [https://github.com/NaverCloudPlatform/naverspeech-sdk-android](https://github.com/NaverCloudPlatform/naverspeech-sdk-android/)를 참고합니다.

**2-3. UX 고려사항**

- 일반적으로 사용자는 음성 인식 버튼을 누르자마자 발화를 시작하려고 할 것입니다.

하지만 음성 인식을 시작하는 recognize() 메서드를 호출하면 음성 인식을 위

한 메모리 할당, 마이크 자원 할당, 음성 인식 서버 접속 및 인증 등의 준비 과정을

수행해야 하기 때문에 사용자의 발화 일부가 누락될 수 있습니다. 따라서, 앱은 모든

준비가 완료된 후 사용자에게 발화해도 좋다는 정보를 전달해야 합니다. 이 방법은

다음과 같이 처리할 수 있습니다.

* 모든 준비가 완료되면 onReady callback 메서드가 호출됩니다.
* onReady callback 메서드가 호출되기 전까지 "준비 중입니다."와 같은 메시지를 표시하거나 준비 중임을 나타내는 UI 표시를 해야 합니다.
* onReady callback 메서드가 호출되면 "이야기해주세요."와 같은 메시지를 표시하거나 사용 가능함을 나타내는 UI를 표시해야 합니다.

- **Note!**

● SpeechRecognitionListener의 onReady, onRecord 등의 callback 메서드는 [Worker Thread](https://developer.android.com/guide/components/processes-and-threads.html)에서 호출되는 메서드이며, Handler에 등록하여 사용해야 합니다.

**2-4. 오류 처리**

- SR API를 사용할 때 다양한 원인으로 오류가 발생할 수 있으며 이때 오류 callback 함

수를 통해 오류 코드가 전달됩니다. 오류 코드를 분석하면 원인을 분석하거나 오류 처

리를 할 수 있습니다. CSR API의 오류 callback 함수가 전달하는 오류는 다음과 같습니다.

| **오류 이름** | **오류 코드** | **설명** |
| --- | --- | --- |
| ERROR\_NETWORK\_INITIALIZE | 10 | 네트워크 자원 초기화 오류 |
| ERROR\_NETWORK\_FINALIZE | 11 | 네트워크 자원 해제 오류 |
| ERROR\_NETWORK\_READ | 12 | 네트워크 데이터 수신 오류. 클라이언트 기기의 네트워크 환경이 느려 Timeout 이 발생하는 경우 주로 발생합니다. |
| ERROR\_NETWORK\_WRITE | 13 | 네트워크 데이터 전송 오류. 클라이언트 기기의 네트워크 환경이 느려 Timeout 이 발생하는 경우 주로 발생합니다. |
| ERROR\_NETWORK\_NACK | 14 | 음성 인식 서버 오류. 느린 네트워크 환경으로 인해 클라이언트가 서버로 음성 패킷을 제시간에 보내지 못하면 서버는 Timeout를 발생시킵니다. 이때 발생하는 오류입니다. |
| ERROR\_INVALID\_PACKET | 15 | 유효하지 않은 패킷 전송으로 인한 오류 |
| ERROR\_AUDIO\_INITIALIZE | 20 | 오디오 자원 초기화 오류. 오디오 사용 권한이 있는지 확인합니다. |
| ERROR\_AUDIO\_FINALIZE | 21 | 오디오 자원 해제 오류 |
| ERROR\_AUDIO\_RECORD | 22 | 음성 입력(녹음) 오류. 오디오 사용 권한이 있는지 확인합니다. |
| ERROR\_SECURITY | 30 | 인증 권한 오류 |
| ERROR\_INVALID\_RESULT | 40 | 인식 결과 오류 |
| ERROR\_TIMEOUT | 41 | 일정 시간 이상 서버로 음성을 전송하지 못하거나, 인식 결과를 받지 못함. |
| ERROR\_NO\_CLIENT\_RUNNING | 42 | 클라이언트가 음성 인식을 수행하지 않는 상황에서 특정 음성 인식 관련 이벤트가 감지됨. |
| ERROR\_UNKNOWN\_EVENT | 50 | 클라이언트 내부에 규정되어 있지 않은 이벤트가 감지됨. |
| ERROR\_VERSION | 60 | 프로토콜 버전 오류 |
| ERROR\_CLIENTINFO | 61 | 클라이언트 정보 오류 |
| ERROR\_SERVER\_POOL | 62 | 음성 인식 가용 서버 부족 |
| ERROR\_SESSION\_EXPIRED | 63 | 음성 인식 서버 세션 만료 |
| ERROR\_SPEECH\_SIZE\_EXCEEDED | 64 | 음성 패킷 사이즈 초과 |
| ERROR\_EXCEED\_TIME\_LIMIT | 65 | 인증용 타임 스탬프(time stamp) 불량 |
| ERROR\_WRONG\_SERVICE\_TYPE | 66 | 올바른 서비스 타입(service type)이 아님. |
| ERROR\_WRONG\_LANGUAGE\_TYPE | 67 | 올바른 언어 타입(language type)이 아님. |
| ERROR\_OPENAPI\_AUTH | 70 | Open API 인증 오류. Client ID와 등록된 package 이름(Android) 또는 Bundle ID 정보(iOS)가 잘못되었을 때 발생합니다. |
| ERROR\_QUOTA\_OVERFLOW | 71 | 정해진 API 호출 제한량(quota)을 모두 소진함. |

위 오류 코드 외에도 다음과 같은 오류가 발생하거나 문의가 있을 수 있습니다.

| **현상 또는 문의** | **원인 또는 해결 방법** |
| --- | --- |
| UnsatifiedLinkError 오류 발생 | CSR API는 armeabi와 armeabi-v7a로 빌드된 라이브러리를 제공합니다. 만약 개발하는 앱에서 사용하는 라이브러리 중 armeabi와 armeabi-v7a를 지원하지 않는 것이 있다면 이 오류가 발생할 수 있습니다. |
| android fatal signal 11 (sigsegv) 오류 발생 | CSR API를 사용하여 음성을 입력받기 전에 우선 자원을 준비해야 합니다. recognize() 메서드를 호출하기 전에 initialize() 메서드가 잘 호출되는지 확인해야 합니다. 또한 release() 메서드도 호출할 수 있어야 합니다. |
| 인식 결과로 ""(null)이 반환됩니다. | 사용자가 매우 작은 목소리로 발성하였거나, 주변 소리로 인해 목소리가 인식되지 않았을 경우 발생할 수 있습니다. 극히 드물게 발생하지만 인식 결과가 null일 때도 예외 처리해주는 것을 권장합니다. |
| 오디오 파일 인식 | CSR API는 오디오 파일 인식을 지원하지 않습니다. |
| 저사양 스마트 폰에서 제대로 동작하지 않습니다. | CSR API는 Android SDK 버전 10 이상과 iOS 버전 8 이상의 기기를 지원하고 있습니다. |

2-5. 구현 예제

* 예제 코드 저장소 : <https://github.com/NaverCloudPlatform/naverspeech-sdk-android>
* 설명
  + [Main Activity](https://github.com/NaverCloudPlatform/naverspeech-sdk-android/blob/ncp/samples/EPDTypeAutoSample/app/src/main/java/com/naver/naverspeech/client/MainActivity.java) 클래스 : SpeechRecognitionListener를 초기화하고, 이후 이벤트를 handleMessage에서 받아 처리합니다.
  + [SpeechRecognitionListener를 상속한 클래스](https://github.com/NaverCloudPlatform/naverspeech-sdk-android/blob/ncp/samples/EPDTypeAutoSample/app/src/main/java/com/naver/naverspeech/client/NaverRecognizer.java) : 음성인식 서버 연결, 음성전달, 인식결과 발생등의 이벤트에 따른 결과 처리 방법 정의합니다.

// 1. Main Activity 클래스

public class MainActivity extends Activity {

private static final String TAG = MainActivity.class.getSimpleName();

private static final String CLIENT\_ID = "YOUR CLIENT ID"; // "내 애플리케이션"에서 Client ID를 확인해서 이곳에 적어주세요.

private RecognitionHandler handler;

private NaverRecognizer naverRecognizer;

private TextView txtResult;

private Button btnStart;

private String mResult;

private AudioWriterPCM writer;

// Handle speech recognition Messages.

private void handleMessage(Message msg) {

switch (msg.what) {

case R.id.clientReady: // 음성인식 준비 가능

txtResult.setText("Connected");

writer = new AudioWriterPCM(Environment.getExternalStorageDirectory().getAbsolutePath() + "/NaverSpeechTest");

writer.open("Test");

break;

case R.id.audioRecording:

writer.write((short[]) msg.obj);

break;

case R.id.partialResult:

mResult = (String) (msg.obj);

txtResult.setText(mResult);

break;

case R.id.finalResult: // 최종 인식 결과

SpeechRecognitionResult speechRecognitionResult = (SpeechRecognitionResult) msg.obj;

List<String> results = speechRecognitionResult.getResults();

StringBuilder strBuf = new StringBuilder();

for(String result : results) {

strBuf.append(result);

strBuf.append("\n");

}

mResult = strBuf.toString();

txtResult.setText(mResult);

break;

case R.id.recognitionError:

if (writer != null) {

writer.close();

}

mResult = "Error code : " + msg.obj.toString();

txtResult.setText(mResult);

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

break;

case R.id.clientInactive:

if (writer != null) {

writer.close();

}

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

break;

}

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

txtResult = (TextView) findViewById(R.id.txt\_result);

btnStart = (Button) findViewById(R.id.btn\_start);

handler = new RecognitionHandler(this);

naverRecognizer = new NaverRecognizer(this, handler, CLIENT\_ID);

btnStart.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if(!naverRecognizer.getSpeechRecognizer().isRunning()) {

mResult = "";

txtResult.setText("Connecting...");

btnStart.setText(R.string.str\_stop);

naverRecognizer.recognize();

} else {

Log.d(TAG, "stop and wait Final Result");

btnStart.setEnabled(false);

naverRecognizer.getSpeechRecognizer().stop();

}

}

});

}

@Override

protected void onStart() {

super.onStart(); // 음성인식 서버 초기화는 여기서

naverRecognizer.getSpeechRecognizer().initialize();

}

@Override

protected void onResume() {

super.onResume();

mResult = "";

txtResult.setText("");

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

}

@Override

protected void onStop() {

super.onStop(); // 음성인식 서버 종료

naverRecognizer.getSpeechRecognizer().release();

}

// Declare handler for handling SpeechRecognizer thread's Messages.

static class RecognitionHandler extends Handler {

private final WeakReference<MainActivity> mActivity;

RecognitionHandler(MainActivity activity) {

mActivity = new WeakReference<MainActivity>(activity);

}

@Override

public void handleMessage(Message msg) {

MainActivity activity = mActivity.get();

if (activity != null) {

activity.handleMessage(msg);

}

}

}

}

// 2. SpeechRecognitionListener를 상속한 클래스

class NaverRecognizer implements SpeechRecognitionListener {

private final static String TAG = NaverRecognizer.class.getSimpleName();

private Handler mHandler;

private SpeechRecognizer mRecognizer;

public NaverRecognizer(Context context, Handler handler, String clientId) {

this.mHandler = handler;

try {

mRecognizer = new SpeechRecognizer(context, clientId);

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

mRecognizer.setSpeechRecognitionListener(this);

}

public SpeechRecognizer getSpeechRecognizer() {

return mRecognizer;

}

public void recognize() {

try {

mRecognizer.recognize(new SpeechConfig(LanguageType.KOREAN, EndPointDetectType.AUTO));

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

}

@Override

@WorkerThread

public void onInactive() {

Message msg = Message.obtain(mHandler, R.id.clientInactive);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onReady() {

Message msg = Message.obtain(mHandler, R.id.clientReady);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onRecord(short[] speech) {

Message msg = Message.obtain(mHandler, R.id.audioRecording, speech);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onPartialResult(String result) {

Message msg = Message.obtain(mHandler, R.id.partialResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetected() {

Log.d(TAG, "Event occurred : EndPointDetected");

}

@Override

@WorkerThread

public void onResult(SpeechRecognitionResult result) {

Message msg = Message.obtain(mHandler, R.id.finalResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onError(int errorCode) {

Message msg = Message.obtain(mHandler, R.id.recognitionError, errorCode);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetectTypeSelected(EndPointDetectType epdType) {

Message msg = Message.obtain(mHandler, R.id.endPointDetectTypeSelected, epdType);

msg.sendToTarget();

}

}

case R.id.recognitionError:

if (writer != null) {

writer.close();

}

mResult = "Error code : " + msg.obj.toString();

txtResult.setText(mResult);

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

break;

case R.id.clientInactive:

if (writer != null) {

writer.close();

}

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

break;

}

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

txtResult = (TextView) findViewById(R.id.txt\_result);

btnStart = (Button) findViewById(R.id.btn\_start);

handler = new RecognitionHandler(this);

naverRecognizer = new NaverRecognizer(this, handler, CLIENT\_ID);

btnStart.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if(!naverRecognizer.getSpeechRecognizer().isRunning()) {

mResult = "";

txtResult.setText("Connecting...");

btnStart.setText(R.string.str\_stop);

naverRecognizer.recognize();

} else {

Log.d(TAG, "stop and wait Final Result");

btnStart.setEnabled(false);

naverRecognizer.getSpeechRecognizer().stop();

}

}

});

}

@Override

protected void onStart() {

super.onStart(); // 음성인식 서버 초기화는 여기서

naverRecognizer.getSpeechRecognizer().initialize();

}

@Override

protected void onResume() {

super.onResume();

mResult = "";

txtResult.setText("");

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

}

@Override

protected void onStop() {

super.onStop(); // 음성인식 서버 종료

naverRecognizer.getSpeechRecognizer().release();

}

// Declare handler for handling SpeechRecognizer thread's Messages.

static class RecognitionHandler extends Handler {

private final WeakReference<MainActivity> mActivity;

RecognitionHandler(MainActivity activity) {

mActivity = new WeakReference<MainActivity>(activity);

}

@Override

public void handleMessage(Message msg) {

MainActivity activity = mActivity.get();

if (activity != null) {

activity.handleMessage(msg);

}

}

}

}

// 2. SpeechRecognitionListener를 상속한 클래스

class NaverRecognizer implements SpeechRecognitionListener {

private final static String TAG = NaverRecognizer.class.getSimpleName();

private Handler mHandler;

private SpeechRecognizer mRecognizer;

public NaverRecognizer(Context context, Handler handler, String clientId) {

this.mHandler = handler;

try {

mRecognizer = new SpeechRecognizer(context, clientId);

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

mRecognizer.setSpeechRecognitionListener(this);

}

public SpeechRecognizer getSpeechRecognizer() {

return mRecognizer;

}

public void recognize() {

try {

mRecognizer.recognize(new SpeechConfig(LanguageType.KOREAN, EndPointDetectType.AUTO));

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

}

@Override

@WorkerThread

public void onInactive() {

Message msg = Message.obtain(mHandler, R.id.clientInactive);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onReady() {

Message msg = Message.obtain(mHandler, R.id.clientReady);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onRecord(short[] speech) {

Message msg = Message.obtain(mHandler, R.id.audioRecording, speech);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onPartialResult(String result) {

Message msg = Message.obtain(mHandler, R.id.partialResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetected() {

Log.d(TAG, "Event occurred : EndPointDetected");

}

@Override

@WorkerThread

public void onResult(SpeechRecognitionResult result) {

Message msg = Message.obtain(mHandler, R.id.finalResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onError(int errorCode) {

Message msg = Message.obtain(mHandler, R.id.recognitionError, errorCode);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetectTypeSelected(EndPointDetectType epdType) {

Message msg = Message.obtain(mHandler, R.id.endPointDetectTypeSelected, epdType);

msg.sendToTarget();

}

}

btnStart.setEnabled(true);

}

@Override

protected void onStop() {

super.onStop(); // 음성인식 서버 종료

naverRecognizer.getSpeechRecognizer().release();

}

// Declare handler for handling SpeechRecognizer thread's Messages.

static class RecognitionHandler extends Handler {

private final WeakReference<MainActivity> mActivity;

RecognitionHandler(MainActivity activity) {

mActivity = new WeakReference<MainActivity>(activity);

}

@Override

public void handleMessage(Message msg) {

MainActivity activity = mActivity.get();

if (activity != null) {

activity.handleMessage(msg);

}

}

}

}

// 2. SpeechRecognitionListener를 상속한 클래스

class NaverRecognizer implements SpeechRecognitionListener {

private final static String TAG = NaverRecognizer.class.getSimpleName();

private Handler mHandler;

private SpeechRecognizer mRecognizer;

public NaverRecognizer(Context context, Handler handler, String clientId) {

this.mHandler = handler;

try {

mRecognizer = new SpeechRecognizer(context, clientId);

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

mRecognizer.setSpeechRecognitionListener(this);

}

public SpeechRecognizer getSpeechRecognizer() {

return mRecognizer;

}

public void recognize() {

try {

mRecognizer.recognize(new SpeechConfig(LanguageType.KOREAN, EndPointDetectType.AUTO));

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

}

@Override

@WorkerThread

public void onInactive() {

Message msg = Message.obtain(mHandler, R.id.clientInactive);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onReady() {

Message msg = Message.obtain(mHandler, R.id.clientReady);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onRecord(short[] speech) {

Message msg = Message.obtain(mHandler, R.id.audioRecording, speech);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onPartialResult(String result) {

Message msg = Message.obtain(mHandler, R.id.partialResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetected() {

Log.d(TAG, "Event occurred : EndPointDetected");

}

@Override

@WorkerThread

public void onResult(SpeechRecognitionResult result) {

Message msg = Message.obtain(mHandler, R.id.finalResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onError(int errorCode) {

Message msg = Message.obtain(mHandler, R.id.recognitionError, errorCode);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetectTypeSelected(EndPointDetectType epdType) {

Message msg = Message.obtain(mHandler, R.id.endPointDetectTypeSelected, epdType);

msg.sendToTarget();

}

}

@Override

@WorkerThread

public void onReady() {

Message msg = Message.obtain(mHandler, R.id.clientReady);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onRecord(short[] speech) {

Message msg = Message.obtain(mHandler, R.id.audioRecording, speech);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onPartialResult(String result) {

Message msg = Message.obtain(mHandler, R.id.partialResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetected() {

Log.d(TAG, "Event occurred : EndPointDetected");

}

@Override

@WorkerThread

public void onResult(SpeechRecognitionResult result) {

Message msg = Message.obtain(mHandler, R.id.finalResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onError(int errorCode) {

Message msg = Message.obtain(mHandler, R.id.recognitionError, errorCode);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetectTypeSelected(EndPointDetectType epdType) {

Message msg = Message.obtain(mHandler, R.id.endPointDetectTypeSelected, epdType);

msg.sendToTarget();

}

}

**3. 음성 재생 관련 API –** 다음 병원 예약이나, 복용알림을 알려주고, 약 추천도 받습니다.

**3-1. CSS**

****

아름다운 성우의 목소리로 글을 읽어주는 애플리케이션을 만들 때 유용한 서비스입니다. 주어진 텍스트를 자연스럽게 말하듯 재생해주어 음성 안내 시스템, 뉴스/책 읽기 서비스 등에 활용할 수 있습니다.

**3-2. API URL**

| **Method** | **Request URI** |
| --- | --- |
| POST | https://naveropenapi.apigw.ntruss.com/voice/v1/tts |

**3-3. 오류 코드**

<JSON>

{

"error": {

"errorCode": "300",

"message": "Not Found Exception"

}

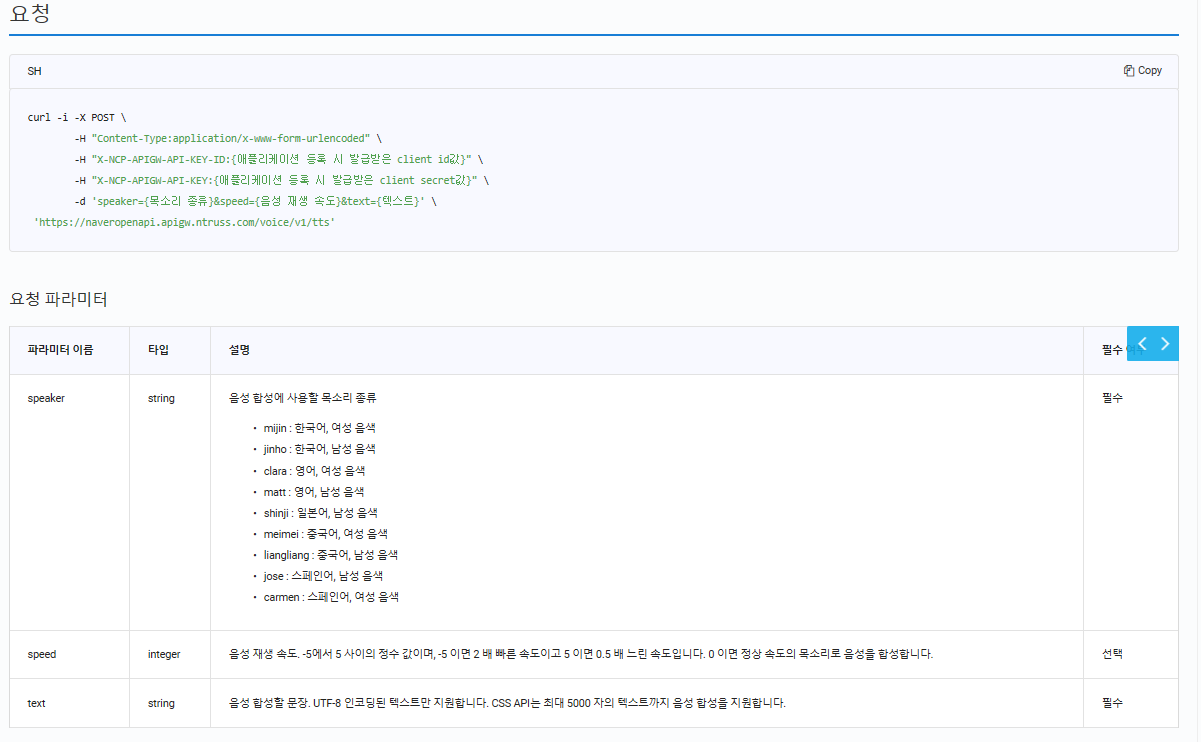
}

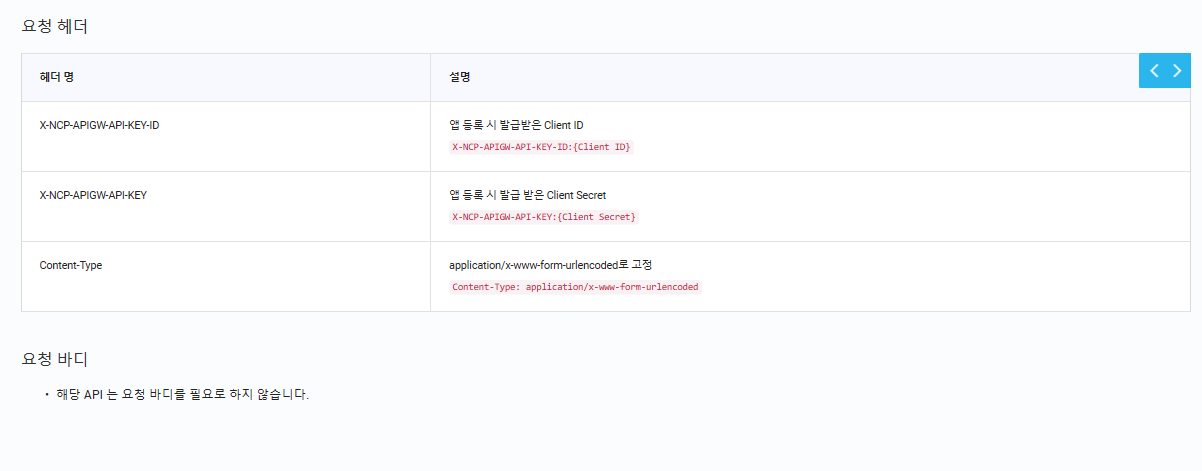
**3-4. 공통 오류**

| **HttpStatusCode** | **ErrorCode** | **ErrorMessage** | **Description** |
| --- | --- | --- | --- |
| 400 | 100 | Bad Request Exception | protocol(https), endocing(UTF-8) 등 request 오류 |
| 401 | 200 | Authentication Failed | 인증실패 |
| 401 | 210 | Permission Denied | 권한없음 |
| 404 | 300 | Not Found Exception | 권한없음 |
| 429 | 400 | Quota Exceeded | Quota 초과 |
| 429 | 410 | Throttle Limited | Rate 초과 |
| 429 | 420 | Rate Limited | Rate 초과 |
| 413 | 430 | Request Entity Too Large | content-length 초과 (10MB) |
| 503 | 500 | Endpoint Error | 엔드포인트 연결오류 |
| 504 | 510 | Endpoint Timeout | 엔드포인트 연결시간 초과 |
| 500 | 900 | Unexpected Error | 예외처리가 안된 오류 |

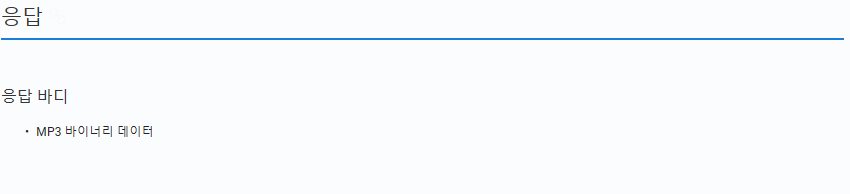
1. **CSS API의 오퍼레이션 tts (** [**https://apidocs.ncloud.com/ko/ai-naver/clova\_speech\_synthesis/tts/**](https://apidocs.ncloud.com/ko/ai-naver/clova_speech_synthesis/tts/) **)**

**4-1. 요청 관련 설명**



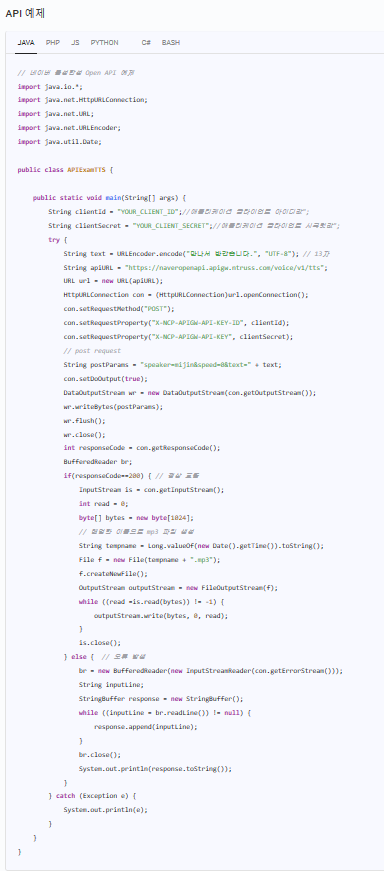


**4-2. 응답 관련 설명**

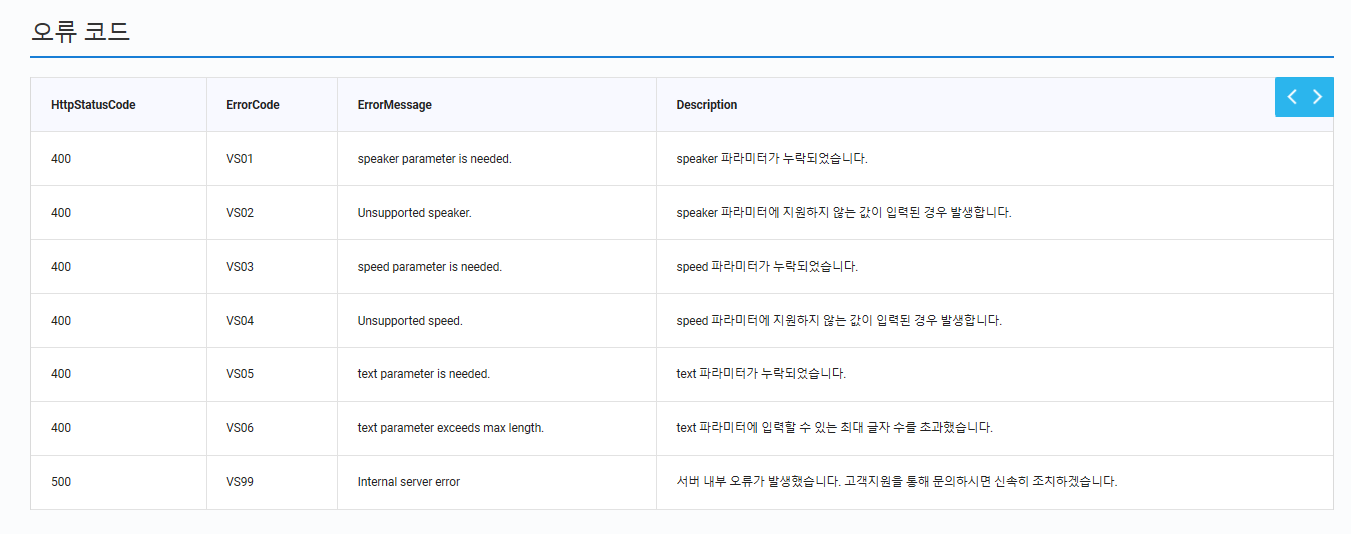


**4-3. 예시**





**4-4. 오류 코드**

****