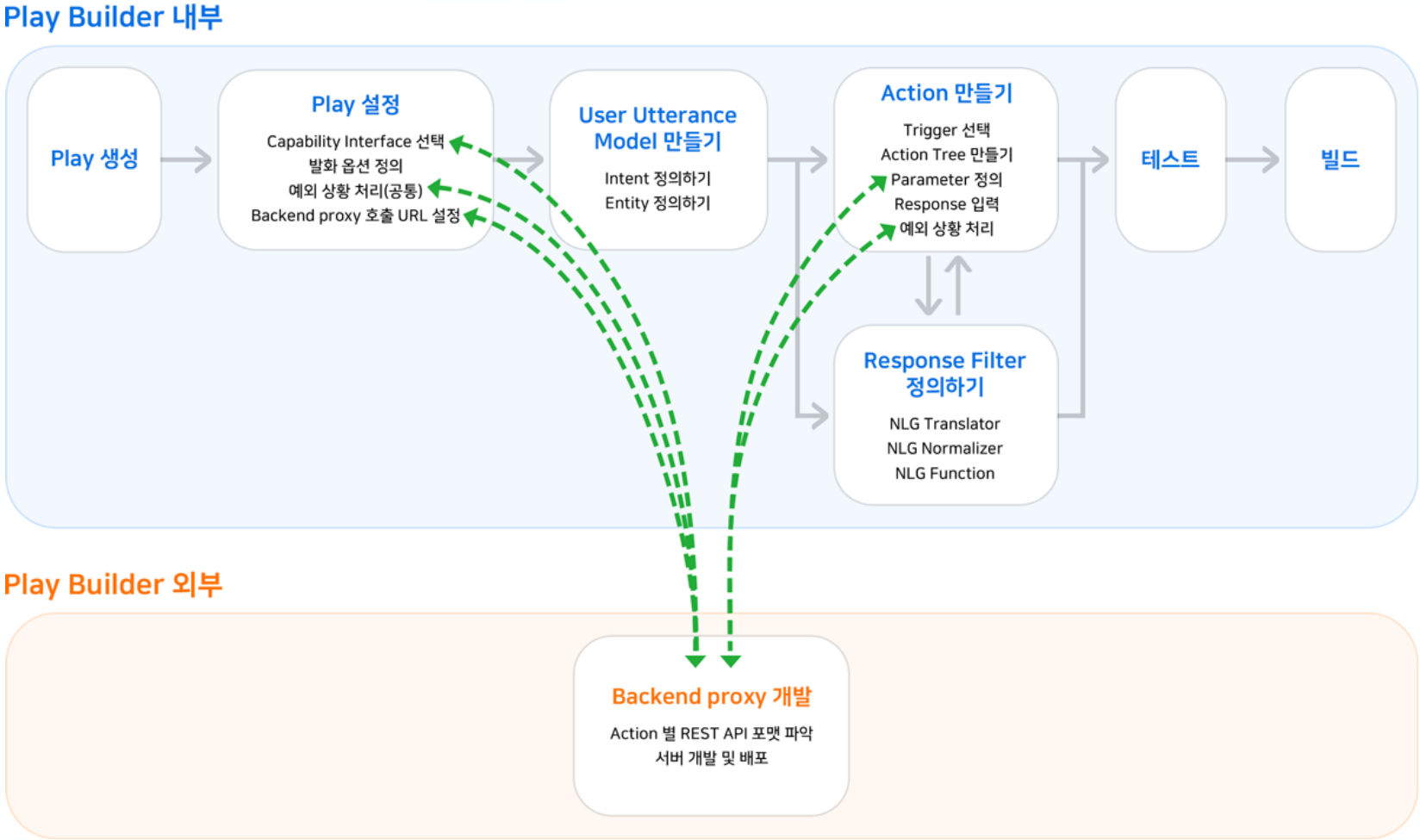
Backend proxy 서버 개발

I. Backend proxy 사용하여 play 만들기

- **Backend proxy (외부 연동 서버) : NUGU 플랫폼에서 REST API를 통해 호출하는 서버**

1. Action의 응답을 정의할 때, 외부 서버로부터 정보를 가져와야 하는 경우
    - 날씨 상태, 지하철 도착 예정 시간, 특정 POI의 전화번호 등
2. 특정 값에 대한 서버의 판단이 필요한 경우
    - 사용자가 말한 날짜가 무슨 요일인지 판단
    - 사용자로부터 2개의 Entity를 받아서 처리해야 할 때, 2개의 Entity의 정합성을 체크하는 경우(2월 31일의 날씨 알려줘, 일본의 뉴욕 시간 알려줘)
3. 서버에서 연산하여 결과를 제공할 수 있는 경우
    - 주사위의 합 : 2 + 3의 결과
4. Directive를 사용하여 디바이스의 기능을 동작시켜야 하는 경우
    - 오디오 재생, 멈춤
5. 예외 상황에 대한 판단이 필요할 때
    - 콘텐츠 서버의 장애
    - 사용자가 지원하지 않는 범위의 정보를 요청했을 때, Intent는 유지하면서 Entity만 다시 받아 처리하고 싶은 경우

I. Backend proxy 사용하여 play 만들기



I. Backend proxy 사용하여 play 만들기

1. Backend proxy 서버 구축
    - REST API를 처리할 수 있는 웹 서버 (개발 언어는 제한 없음)
    - Backend proxy 개발에 대한 자세한 내용은 외부 연동 서버(Backend proxy) 문서 참고

2. Backend proxy를 Play와 연결
    - Play Builder에서 해당 Play의 General → 외부 서버 연결 정보 페이지에서 Backend proxy의 Web URL을 입력
        - Web URL 입력 시 http:// 또는 https://를 포함하여 작성
    - Backend proxy과의 통신이 실패했을 때 사용자에게 전달할 메시지를 '연결 실패 시 prompt'에 입력
    - Backend proxy와 Play를 연결하는 자세한 방법은 Play 설정하기 문서 참고

3. Play 개발자는 Backend proxy 개발자와 다음 사항을 논의해야 함

   ① 어떤 Utterance Parameter를 정의하였고, 해당 Utterance Parameter는 어떠한 값들이 전달이 될 것인가를 논의

   ② Entity를 정규화 한다면 대표 값이 Utterance Parameter에 담겨서 전달이 될 것이므로 해당 대표 값을 논의

   ③ 어떠한 Backend Parameter가 필요하고, 해당 parameter에는 어떤 값이 담기기를 기대하는지를 논의

   ④ Play를 만들어가면서 발생되는 예외 상황들에 대해 안내를 하고 각 상황 별 code(Exception Code)를 논의

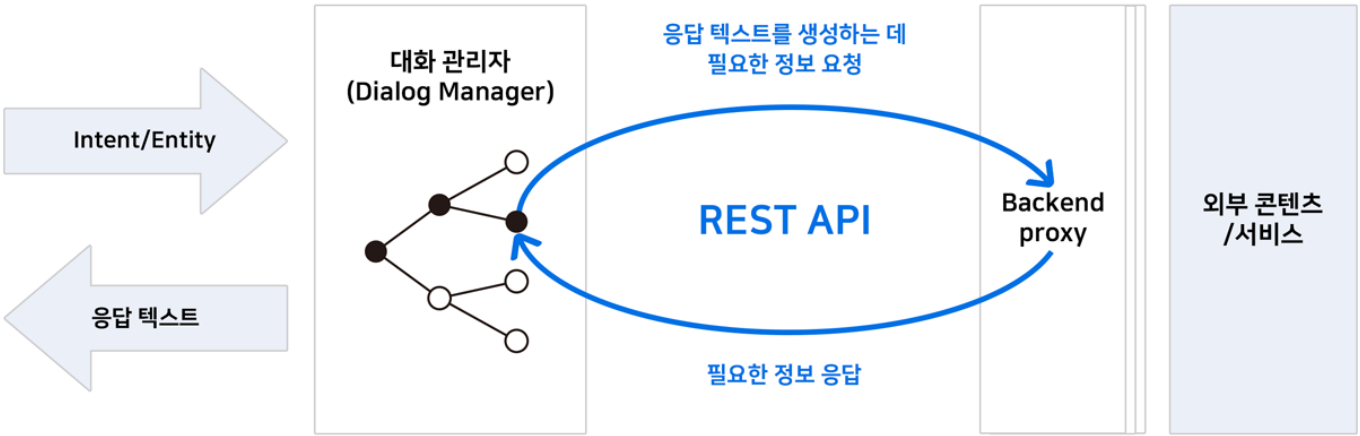
   ⑤ Capability Interface를 사용하는 경우 어떤 Intent에서 어떻게 Directive를 내보낼 것인지 논의

4. Backend proxy 개발자와 논의한 바탕으로 Backend Parameter를 활용하고, 예외 상황 Prompt도 입력

   ▪ 예외 처리에 대한 자세한 내용은 예외 상황 관리 문서 참고

I. Backend proxy 사용하여 play 만들기

- NUGU 플랫폼의 Dialog Manager는 지정된 포맷(Backend proxy API 규격)으로 Request
  - 외부 서비스의 REST API 포맷이 Backend proxy API 규격과 다르다면 포맷을 변환해주기 위한 Backend proxy 서버를 개발
  - Play Builder를 통해 Play를 만드는 과정에서 정의한 파라미터와 Backend proxy API Reference에서 제공하는 규격을 사용하여 직접 개발
- NUGU 플랫폼에서는 Backend proxy를 구축할 수 있는 클라우드 환경을 제공하고 있지 않음
  - Play를 개발하는 곳에서 직접 구축을 하거나 클라우드 서비스를 이용



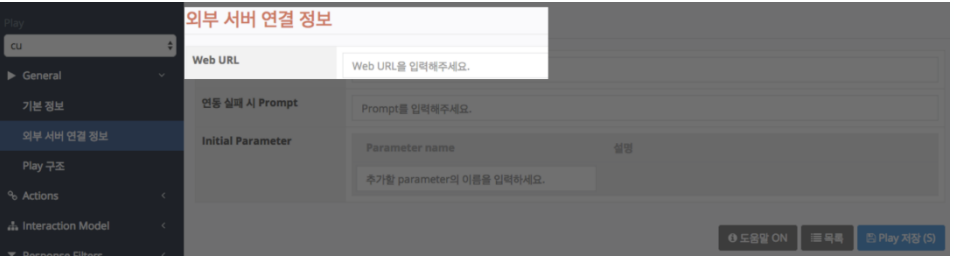
I. Backend proxy 사용하여 play 만들기

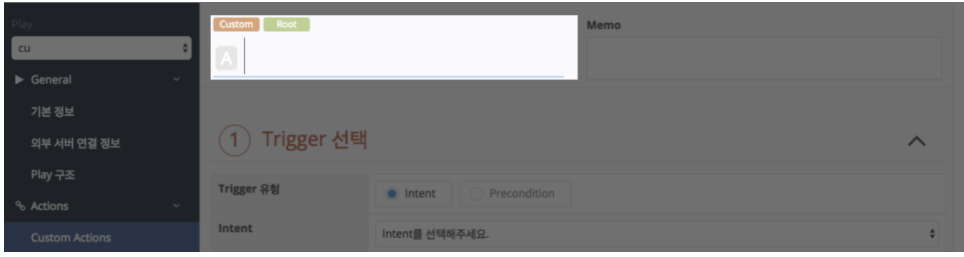
- Backend proxy API 규격은 다음과 같은 정보를 Backend proxy 서버로 전달
  - 다음과 같은 정보를 바탕으로 특정 사용자가 전달한 요청에 대해 적절한 정보를 Response

정보	설명
Action 이름	Backend proxy에서 처리해야 하는 요청을 구분하는 데 사용 어떠한 Action이 Backend proxy를 호출했는지 확인
Parameters	Play에서 정의된 Parameter들이 전달 Utterance Parameter에는 Play 사용자의 실제 발화에 담긴 Entity 혹은 그 Entity가 정규화된 값이 "value"로 전달 Backend Parameter는 "value"를 담아서 Play로 전달하게 될 Parameter이며 value는 "null"로 전달 Utterance/Backend Parameter를 구분할 수 없으므로, Play에서 어떻게 정의를 했는지 파악한 후 구현
Context 정보	사용자 식별 token, 디바이스 상태 정보 등이 전달
Event 정보	디바이스에서 발생한 Event 정보

I. Backend proxy 사용하여 play 만들기

- REST API 호출은 Backend proxy를 사용하도록 지정한 Action에서만 이루어지며, 각 Action 별로 고유한 REST API URL이 결정
- REST API URL 생성 규칙
  - Play Builder 〉 General 〉 외부 서버 연결 정보 〉 Web URL + Play Builder 〉 Actions 〉 Action Name
- 예를 들어 외부 서버 연결 정보의 Web URL에 http://backend_proxy.nugu.com를 설정하고, Action 이름을 "playMusic"으로 설정
  - 해당 Action을 처리하는 REST API URL은 http://backend_proxy.nugu.com/playMusic이 됩니다.





I. Backend proxy 사용하여 play 만들기

Request Body 스펙

```
{
    "version": "2.0",
    "action": {
        "actionName": "{{string}}",
        "parameters": {
            KEY: {
                "type": "{{string}}",
                "value": VALUE
            }
        }
    },
  "event": {
    "type": "{{string}}"
  },
    "context": {
        "session": {
            "accessToken": "{{string}}"
        },
        "device": {
            "type": "{{string}}",
            "state": {
                KEY: VALUE
            }
        },
        "supportedInterfaces": {
            "AudioPlayer": {
                "playerActivity": "PLAYING",
                "token": "string value",
                "offsetInMilliseconds": 100000
            }
        },
        "privatePlay" : { } // reserved
    }
}
```

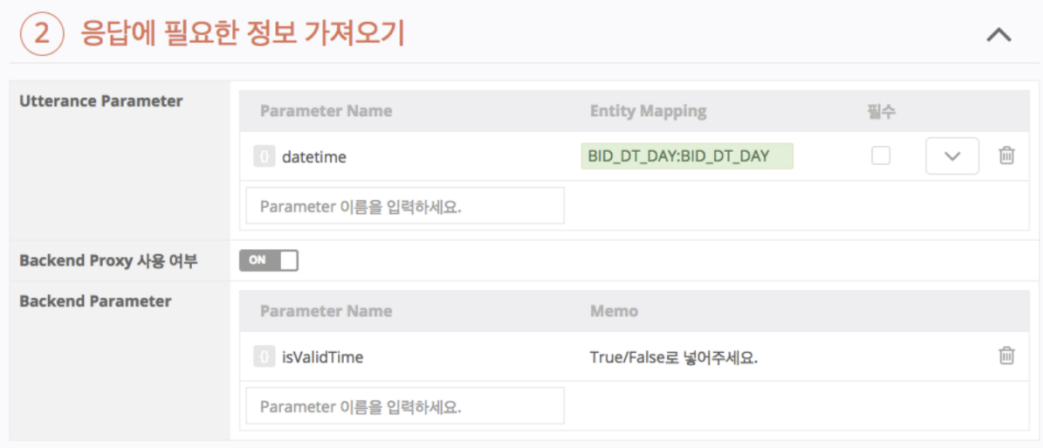
Parameter	Type	Mandatory	Description
version	string	Y	Backend proxy API 버전을 표시
action	json		
action.actionName	string	Y	현재 요청하는 Action의 이름
action.parameters	string	Y	Action에서 설정된 파라미터로 Play Builder에서 설정한 내용을 포함 (단, 값이 null인 경우 요청에서 제외 요청에서 생략되었더라도 Backend parameter를 응답 값으로 포함) KEY – Play Builder에서 Action 내에 정의한 parameter 이름 type – 사용자 발화에서 분석된 Entity인 경우 Play Builder에서 설정한 Entity의 타입 value – 파라미터의 값으로 string 타입
event	json	Y	
event.type	string	Y	디바이스에서 발생한 event의 종류를 나타내며, 이 값에 따라 event의 데이터가 달라집니다. (Capability Interfaces 참조)
context	json	Y	
context.session	json	Y	
context.session.id	string	Y	대화가 유지되는 동안의 유효한 키 값
context.session.isNew	boolean	Y	대화의 처음을 알려주는 값
context.session.accessToken	string	N	OAuth 2에 사용되는 인증 token
context.session.isPlay BuilderReques t	bool	N	Play Builder에서 테스트용으로 전달한 요청임을 의미 (기본값: false)
context.device	json	Y	
context.device.type	string	Y	현재 사용 중인 디바이스 종류
context.device.state	json	Y	디바이스의 상태를 나타내는 값 (현재는 정의된 것이 없음)
context.supportedInterfaces	json	Y	개발한 Play가 특정 Capability Interface를 지원하는 경우 각 Interface별로 상태 정보를 표시

- Request의 Body로 전달되는 JSON 포맷의 데이터는 Backend proxy API Reference를 참조
  - JSON 데이터는 임의의 필드가 추가 될 수 있으므로, 이에 대한 영향이 없도록 구현
- Request Body에 정의된 필드 중 Play Builder에 설정한 값들에 의해 결정되는 필드는 action.parameters

필드명	생성 규칙
KEY	Play Builder의 "응답에 필요한 정보 가져오기" 화면에서 다음의 두 위치에 설정된 모든 파라미터가 포함되어야 함 - Utterance Parameter의 Parameter Name - Backend Parameter의 Parameter Name
type	Utterance Parameter에만 적용되며, Play Builder의 Entity Mapping에 설정된 값 중 ":" 앞의 값만 전송 Backend Parameter의 경우에는 Entity Mapping을 설정하지 않기 때문에 이 필드는 사용되지 않음
value	실제 해당 파라미터에 할당된 값을 전송 Utterance Parameter의 경우 필수가 체크되지 않은 파라미터는 값이 있을 수도 있고 없을 수도 있음 Backend Parameter의 경우에도 Backend proxy가 몇 번 호출되느냐에 따라 값이 있을 수도 있고 없을 수도 있음

I. Backend proxy 사용하여 play 만들기

- Backend Parameter의 경우 Backend proxy에서 처리한 결과를 가져오는 용도로 사용되기 때문에 Request Body에는 null을 갖게 됨
- Backend Parameter에 어떤 값을 채워줄지는 Play Builder에서 설정한 Action의 용도에 따라 다름
  - Play Builder 작성자와 Backend proxy 개발자 간에 어떻게 처리해야 할지 정확하게 내용을 공유하고 있어야 함




```
"action": {
  "actionName": "action이름",
  "parameters": {
    "datetime": {
      "type": "BID_DT_DAY",
      "value": "오늘"
    },
    "isValidTime": {
      "type": null,
      "value": null
    }
  }
}
```

I. Backend proxy 사용하여 play 만들기

Response Body 스펙

```
{
    "version": "2.0",
    "resultCode": "OK",
    "output": {
      "datetime": "오늘",
      KEY1: VALUE1,
      KEY2: VALUE2,
      ...
    },
    "directives": [
      {
        "type": "AudioPlayer.Play",
        "audioItem": {
            "stream": {
                "url": "{{STRING}}",
                "offsetInMilliseconds": {{LONG}},
                "progressReport": {
                    "progressReportDelayInMilliseconds": {{LONG}},
                    "progressReportIntervalInMilliseconds": {{LONG}}
                },
                "token": "{{STRING}}",
                "expectedPreviousToken": "{{STRING}}"
            },
            "metadata": { } // reserved
        }
      }
    ]
}
```

Parameter	Type	Mandatory	Description
version	string	Y	Backend proxy API 버전을 표시
resultCode	string	Y	"OK" – **성공인 경우 사용하는 값으로 다른 값을 전송하면 성공이 아닌 것으로 처리하기 때문에 주의해야 함** 성공이 아닌 경우는 아래 예외 처리에서 설정된 Result Code(Exception Code)값 전송 – PlayBuider의 General 〉 기본정보 페이지의 예외 처리 – Action 〉 Custom Actions〉 선택한 Action의 예외 처리
output	json	Y	Request에서 전송한 action.parameters의 KEY:VALUE를 처리한 결과를 전송 Request의 모든 KEY:VALUE가 동일하게 나와야 함 – VALUE는 Request의 값과 같거나 다를 수 있음 변경되지 않은 VALUE들은 Request의 값을 그대로 써주어야 함 KEY – Request의 action.parameters에 정의된 KEY VALUE – backend proxy에서 처리한 결과
directives	json	N	특정 Capability Interface를 지원하는 Play에서 Directive를 전송하는 경우에 이 필드를 통해 전송
			각 Capability Interface의 Directive 포맷은 해당 Capability Interface 규격을 참조

- REST API 요청에 대한 응답으로 생성되는 Body는 Request Body의 action.parameters에 전달했던 모든 KEY를 동일하게 Response에 전달해주어야함

  1. Request의 action.parameters에 사용된 KEY는 모두 Response의 "output" 내의 KEY로 정의되어야 함
  2. "output" 내의 KEY는 Request와 다르게 "type", "value" 필드를 갖지 않고, string 타입의 값(VALUE)만을 포함
  3. Utterance Parameter와 Backend Parameter 모두 Backend proxy에 요청을 한 뒤에 값이 바뀔 수 있음
  4. Request에서 값을 갖거나 null일 수 있으며, Response에서는 요청 값을 그대로 갖거나 변경되거나 null이 될 수 있음

- Response에서 모든 값을 그대로 똑같이 포함하는 이유는 입출력이 명확히 구분되지 않기 때문
  - 사용자 발화에 의해 설정된 Utterance Parameter의 경우 Backend proxy에서 값을 바꾼 뒤에 응답 텍스트에 사용될 수도 있음
  - 규격만을 정확히 따른다면 어떻게 사용하든 크게 문제되지 않음

- "output" 필드 외에 "resultCode"가 정의되어 있음

- Play Builder는 Backend proxy 호출 시 발생할 수 있는 다양한 예외 상황마다 적절히 대응할 수 있도록 설정
  - "Exception Code"와 각 상황에서의 Exception Prompt 또는 사용자에게 오류 상황을 알려주고 필요한 Parameter를 다시 물어볼 수 있음
  - 이때의 각 상황은 Backend proxy의 "resultCode"로 전달되는 값과 Play Builder에서 설정한 "Exception Code"를 매칭하여 동작을 결정

- 따라서 Response Body로 전달되는 JSON 포맷에서 "resultCode"에 올 수 있는 값의 종류와 수행되는 로직은 다음과 같음
  - "OK" → 성공일 경우 Backend proxy는 고정된 값을 보내줘야 함 (이 값 외의 모든 경우(case)는 예외 처리)
  - Exception Code → Play Builder에서 정의한 Exception Code 값으로 Play Builder에서 설정한 Prompt 수행
  - 이외의 모든 값 ("", null 포함) → Play Builder 〉 외부 서버 연결 정보 〉 연동 실패 시 Prompt 영역에서 지정한 기본(Default) Prompt 수행

- directive에는 Play가 지원하는 Capability Interface에서 정의한 Directive가 올 수 있음
  - Play는 임의 개수의 Capability Interface를 지원하도록 설정할 수 있고, Play가 지원하는 Capability Interface의 모든 directive가 Response Body 내에 포함될 수 있음
- 어떤 Play가 2개의 Capability Interface를 지원한다면 "directive" 필드에는 0개, 1개, 2개의 Directive가 포함될 수 있음

- 서비스 정상 여부를 확인하기 위해 다음의 /health url을 다음과 같이 구현해야 함
  - NUGU developers에서는 이 URL을 주기적으로 요청해서 서버의 정상 여부를 판단
  - 정상적으로 서비스가 가능하면 HTTP Status code를 "200 OK"로 리턴 (결과 텍스트는 OK 등 아무 문자나 리턴해도 됨)
- 만약 서비스에 문제가 있을 경우에는 "500 Internal Server Error" 등 200 이외의 HTTP Status Code를 리턴하면 됨
- 심사 요청 시 /health url이 정상 동작해야 하며, /health url에서 200 이외의 상태가 오래 지속되면 서비스가 직권 중지될 수 있으므로 유의

```
GET /health HTTP/1.1
Accept: */*


HTTP/1.1 200 OK
Content-Length: 2
OK
```

End of Document