SAMSUNG SW ACADEMY FOR YOUTH

對外秘

자기주도 PJT

코딩 테스트 채점 서버 만들기

목차

對外秘

1. 과제 개요

본 과제는 코딩 테스트 채점 서버를 만들어보면서 API 서버를 구축하는 방법을 익히고, 기본적인 REST 통신을 사용해보는 과제입니다. 1 학기 교육 과정에서 많이 사용해봐서 익숙한 코딩 테스트 환경을 일부 구현해볼 수 있으며, 그동안 유저 입장에서 제출하기만 했던 코딩 테스트에 대해서 다시 한 번 생각해볼 수 있는 기회가 될 수 있습니다.

SSAFY 의 2 학기는 자기주도 학습을 기반으로 하고 있습니다. 요구사항은 큰 방향성과 약간의 힌트만을 담고 있을 뿐, 구현에 필요한 모든 단계를 알려주고 있지 않습니다. 개발자는 기획이나 영업으로부터 넘겨받은 추상적인 요구사항을 구체화하고 문제를 해결하는 사람입니다. 요구사항에 모든 정답이 담겨 있지 않으니 스스로 문제를 찾고 해결해보세요.

여러분들은 1 학기 동안 익힌 기술 스택을 기반으로 웹서버를 구축하게 되고, 정답 코드를 포함하고 있는 JSON 형태의 제출 데이터를 받아서 코드의 정답 여부를 판단해서 리턴값을 되돌려주게 됩니다. 서버는 Spring Boot 나 Django 혹은 기타 사용 가능한 프레임워크를 활용하시면 되고, Spring Boot 로 구현하는 경우에는 Java 코드를 제출 받고 Django 로 구현하는 경우에는 Python 코드를 제출 받아서 채점하면 됩니다. 하루라는 짧은 시간에 구현을 완료해야 하기 때문에 프론트엔드의 구현은 생략합니다. 과제 진행 후에 시간이 남거나 포트폴리오화를 원하는 교육생들은 자율적으로 추가 구현을 진행하면 됩니다.

기능 명세의 요구사항 4,5,6 번은 심화 과제입니다. 서버 구현이 익숙하지 않은 경우에는 하루라는 짧은 시간에 모든 요구 사항을 구현하기 어려울 수 있습니다. 하루 안에 모든 요구사항의 구현이 어려운 경우 1,2,3 번 명세까지 구현하고 채점 결과는 서버에 로그 형태로 남기면 됩니다.

2. 과제 목표

1) **제출한 데이터를 수신하고 오류 코드를 리턴하는 API 서버 구현**

언어와 프레임워크에 맞는 REST 통신을 담당하는 코드를 구현합니다. 사용자로부터 제출 받은 JSON 형태의 데이터를 수신할 수 있어야 합니다.

2) 채점 데이터 프리셋 생성

사용자로부터 받은 코드를 검증할 채점 데이터 프리셋을 생성합니다. 문제의 제한 조건을 고려해서 랜덤 input 을 생성하거나 경계값을 포함한 input 을 생성합니다. 그에 맞는 output 을 생성해서 정답 데이터로 활용합니다

.

3) 정답 여부 판별

사용자로부터 받은 코드를 컴파일/실행하고, input 에 대한 output 이 정확하게 나오는지를 검증합니다. 발생하는 오류들이 있는지 확인합니다. 사용자로부터 받은 JSON 에서 코드의 문자열을 추출해서 파일 형태로 변환해야 합니다.

4) 주어진 제한 시간에 맞는 채점

문제의 제한 조건으로 주어진 시간을 초과하게 되면 채점을 종료하고 시간 초과로 판정합니다.

5) 상황에 맞는 코드를 리턴

채점 결과에 맞는 코드를 리턴합니다. 오류가 발생했을 때는 상황에 맞는 오류 코드를 함께 리턴합니다. 컴파일 오류나 시간 초과 등 발생할 수 있는 상황에 맞는 오류 코드를 리턴해줘야 사용자들이 그에 맞게 코드를 수정할 수 있기 때문입니다.

3. 필수 지식 학습

코딩 테스트 채점 서버 구현을 위해서는 기본적인 REST 통신의 개념에 대해 이해해야 합니다. REST 통신에 대한 이해를 마치면, 평소에 사용하던 친숙한 언어와 프레임워크를 이용해서 API 서버를 구현합니다. 이미 1 학기 과정을 통해서 학습한 내용이겠지만, 프레임워크에 대해서 학습할 때는 블로그나 유튜브 등을 통해서 학습하기보다 반드시 레퍼런스 문서를 통해서 학습하실 것을 권장합니다.

API 서버의 이해와 구현을 위한 참고 자료

구분	제목	링크
이해	REST 통신이란?	https://gmlwjd9405.github.io/2018/09/21/rest-and-restful.html
구현	Python 을 이용한 API 서버 구현	https://www.django-rest-framework.org/api-guide/routers/
구현	Java 와 Spring Boot 를 이용한 API 서버 구현	https://spring.io/guides/gs/rest-service/
구현	Java SecureRandom	https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html
구현	Python Random numbers	https://docs.python.org/3/library/secrets.html#random-numbers
구현	Curl 을 이용한 REST 통신	https://www.lesstif.com/pages/viewpage.action?pageId=14745703

4. 기능 명세

1. 기능/과제 목록

Req.	Category
1	API 서버 구현
2	채점 데이터 프리셋 생성
3	정답 여부 판별과 채점 모듈 구현
4	주어진 제한 시간에 맞는 채점**(심화)**
5	상황에 맞는 상태 코드 리턴**(심화)**
6	정답 데이터를 제출하고 원하는 리턴이 나오는지 검증**(심화)**

2. 기능/과제 상세

Req. 1. API 서버 구현

Req. 1-1	API 서버 구현

기능 상세	'3. 필수 지식 학습' 부분을 참조해서 친숙한 언어와 프레임워크를 활용한 API 서버를 구축해본다.
	
	1. 사용자의 데이터는 Request Body 에 담겨 있으며, JSON 형식이다.
	
	2. 1.의 JSON 에서 정답 코드를 담은 문자열을 추출한다.
	
	3. 요청은 POST 방식으로 보내지기 때문에 서버에서도 이에 맞는 적절한 처리가 필요하다.
	
	4. 요청으로 들어오는 객체가 담고 있는 정보와 변수명 등은 자유롭게 정해도 좋고, 아래의 객체를 참고하자면 수신한 JSON 으로부터 answer 를 추출해내면 될 것이다. 추출한 정답 코드는 Solution.java 와 같은 형태로 저장하면 된다.
	
	`{`
	`  user : "서울 3 반김삼성",`
	`  answer : "some awesome code"`
	`}`
	
	5. 스스로 API 서버를 구축하기 어렵다면 3.의 공식 문서 링크를 참조해서 한 단계씩 따라해본다. 잘 알고 있다고 하더라도 공식 문서를 보면서 공부한 적이 없다면 이번 기회에 살펴보자.

Django SimpleRouter

Here's an example of a simple URL conf, that uses `SimpleRouter`.

```
from rest_framework import routers

router = routers.SimpleRouter()
router.register(r'users', UserViewSet)
router.register(r'accounts', AccountViewSet)
urlpatterns = router.urls
```

There are two mandatory arguments to the `register()` method:

- `prefix` - The URL prefix to use for this set of routes.
- `viewset` - The viewset class.

Optionally, you may also specify an additional argument:

- `basename` - The base to use for the URL names that are created. If unset the basename will be automatically generated based on the `queryset` attribute of the viewset, if it has one. Note that if the viewset does not include a `queryset` attribute then you must set `basename` when registering the viewset.

The example above would generate the following URL patterns:

- URL pattern: `^users/$` Name: `'user-list'`
- URL pattern: `^users/{pk}/$` Name: `'user-detail'`
- URL pattern: `^accounts/$` Name: `'account-list'`
- URL pattern: `^accounts/{pk}/$` Name: `'account-detail'`

Java RestController

COPY

```
package com.example.restservice;

import java.util.concurrent.atomic.AtomicLong;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GreetingController {

        private static final String template = "Hello, %s!";
        private final AtomicLong counter = new AtomicLong();

        @GetMapping("/greeting")
        public Greeting greeting(@RequestParam(value = "name", defaultValue = "World")
String name) {
                return new Greeting(counter.incrementAndGet(), String.format(template,
name));
        }
}
```

자세한 내용은 링크에 있는 공식 문서를 참조하라.

Req. 2. 채점 데이터 프리셋 생성

Req. 2-1	채점 데이터 프리셋 생성
기능 상세	채점 데이터 프리셋을 생성한다. 사전에 정해진 문제의 조건에 따라서 input.txt 와 같은 채점 데이터 파일을 생성한다. 1. 문제 풀이에 활용할 코딩 테스트 문제를 선정한다. 어떤 문제건 상관 없다.

	2. 해당 문제의 조건을 살펴보고 채점 데이터로 사용할 프리셋을 생성한다. 3. 채점 데이터를 만드는 가장 쉬운 방법은 조건에 맞는 input 을 랜덤 함수를 이용해서 생성하는 것이다. 실제로 삼성 SW 상시 테스트 B 형 이상의 난이도 높은 코딩 테스트의 경우에는 스스로 검증하기 위해 채점 데이터를 만들어서 사용하는 경우도 있다. 4. 랜덤 함수만을 이용해서 생성한 채점 데이터의 경우에는 매우 많은 숫자를 만들어낸다면 코너 케이스들도 검증이 가능할 수도 있지만, 어느 정도는 의도를 가지고 코너 케이스를 검증할 수 있는 채점 데이터를 직접 만들 필요가 있다. 5. 제출한 코드에서 표준 입출력을 처리하기 어렵다면 파일 입출력으로 제출할 것을 가정하고 input.txt 를 읽도록 처리해도 된다. 6. 생성한 input.txt 에 맞는 output 을 만들어서 채점 시에 활용한다. output 은 text 파일 형태로 만들어서 필요할 때마다 읽어서 제출된 데이터와 비교해도 되고, 원한다면 DB 에 넣어두고 사용해도 좋다.

Req. 3. 정답 여부 판별과 채점 모듈 구현

Req. 3-1	정답 여부 판별과 채점 모듈 구현
기능 상세	이 프로젝트에서 가장 어려운 부분이고, 가장 핵심적인 부분이다. 사용자로부터 받은 코드를 채점 데이터를 이용해서 검증하고 정답 여부를 판별한다. 1. 사용자로부터 받은 코드 문자열을 파일 형태로 저장한다. 컴파일이 필요한 경우 컴파일을 진행한다. Java 나 Python 실행 환경이 사전에 구축되어 있어야 한다. 특히 Java 의 경우에는 컴파일 환경이 갖추어져 있어야 한다. 2. 컴파일이 완료되면 코드를 실행한다. 실행하는 방법은 여러 가지 방법이 있으니 자유롭게 선택하면 된다. 파일을 직접 실행하는 방법도 있고, Command line executor 와 같은 키워드로 검색하면 충분한 힌트를 얻을 수 있을 것이다. 3. 별도의 채점 모듈을 만들어놓고 사용자가 제출한 코드를 그 모듈에 포함시켜서 채점하는 형태를 권장한다. 사전에 적절한 폴더 구조를 갖춰두고 input.txt 를 로드하는 형태로 구현하자. 4. 발생할 수 있는 예외에 대해서는 미리 생각해보고 예외 처리를 해둔다. 예외 발생 시에는 적절한 예외 코드를 리턴한다.

Req. 4. 주어진 제한 시간에 맞는 채점(심화)

Req. 4-1	주어진 제한 시간에 맞는 채점
기능 상세	사용자가 제출한 코드가 제한 시간을 초과한 경우, 계속해서 서버의 리소스를 소모하면서 코드를 실행할 필요는 없다. 1. 시간을 초과한 경우에는 작업을 중단하고 시간 초과 코드를 리턴해준다. 방법은 여러 가지가 있으나 Thread 와 Timer 를 활용할 수 있다. 2. API 서버에서 Thread 를 직접 실행하는 방법보다 별도의 채점 모듈을 만들어서 API 서버에서는 그 모듈을 호출하고, 모듈에서 Thread 를 만들어 실행한 뒤 제한 시간을

	초과하면 모듈에서 Thread 를 중단하는 방식이 보다 구현하기 쉬울 것이다. Java 의 경우 사용자가 제출한 코드를 포함하고 있는 Solution.java 와 채점 모듈을 함께 컴파일해서 jar 형태로 빌드하고 실행시켜야 할 수 있다.

Req. 5. 상황에 맞는 상태 코드 리턴(심화)

Req. 5-1	상황에 맞는 상태 코드 리턴
기능 상세	사용자는 최종적으로 자신의 제출한 코드에 대한 상태 코드를 리턴 받는다. 1. 발생할 수 있는 상황은 매우 다양하기 때문에(ex, 컴파일 에러, 스택오버플로우, 시간 초과 등) 상황에 맞는 상태 코드를 리턴해준다.

Req. 6. 정답 데이터를 제출하고 원하는 리턴이 나오는지 검증(심화)

Req. 6-1	정답 데이터를 제출하고 원하는 리턴이 나오는지 검증
기능 상세	최종적으로 실제로 정답 데이터를 제출하고 예상대로 동작하는지 검증한다. 1. curl 이나 postman 등의 도구를 이용해서 JSON 형태의 정답 데이터를 제출하고 제출한 값에 맞는 리턴이 오는지 확인한다. JSON 형태의 데이터를 보낼 때 헤더에 포함해야 하는 것이 있다는 것을 주의하자.

5. 산출물 제출

- https://lab.ssafy.com/s10-study/self-project/ 의 "산출물 제출 가이드".docx 참조
- 작성 내용

산출물 제출	생성한 프로젝트를 gitlab 에 제출한다. 채점 데이터 프리셋은 프로젝트 내의 src 폴더 하위의 input, output 폴더에 보관한다. 전체 요구사항을 모두 구현하지 못하더라도 마크다운 문서에 요구사항 별로 구현한 내용까지 기록한다.

6. 추천 학습 방법

반드시 레퍼런스 문서를 통해서 학습하실 것을 권장합니다. 특히 Spring 과 Django 를 이용해서 API 서버를 구현하는 과정은 유튜브나 블로그의 글을 보기보다 레퍼런스 문서를 통해서 학습해야 합니다. 레퍼런스 문서는 프레임워크를 만든 사람들이 직접 작성한 문서이고, 그들의 생각과 철학을 담고 있기 때문입니다.