Programming Project Portfolio

Jang Jisoo

Github URL – https://github.com/JisooJang

Programming Project Portfolio

1) Chatbot-project ‘서울이랑’

Programming Project Portfolio

Chatbot-project '서울이랑'

서울이랑은 서울시의 종합적인 생활 편의 서비스를 제공하기 위해
교통 정보, 날씨 정보, 맛집 정보, 개인별 맞춤 정보를
카카오톡 플러스친구를 통해 제공하는 챗봇 서비스

프로젝트 특징

대중교통(버스 지하철) 정보, 날씨(미세먼지 및 기온) 정보, 맛집 정보 제공
맛집정보는 사용자가 선택한 기준 지역에 따라 네이버 검색정보를 이용하여 댓글/리뷰순으로 맛집 검색 가능
사용자의 즐겨 찾는 장소를 데이터베이스에 저장하고, 그 데이터를 바탕으로 사용자 맞춤 서비스
(사용자가 자주 찾는 교통정보, 날씨정보, 맛집 정보)를 제공

수행 기간 : 2017.03 ~ 2017.09
참여 인원 : 서버 담당 팀원 4명

Programming Project Portfolio

Chatbot-project '서울이랑'

프로젝트 적용 기술 및 사용 툴

- spring boot

spring boot는 스프링 프레임워크를 사용하는 프로젝트를 간편하게 셋업할 수 있는 스프링 프레임워크의 서브프로젝트이다.
일반 Spring MVC 프로젝트에 비해 초기에 객체 간 설정 작업이 생략 되어 속도 있게 진행이 가능하였다.
spring boot를 사용하여 RestController 서버를 구축하고 카카오 플러스 친구API 와 연동하는데 사용하였다.

- insomnia, postman

REST/HTTP API 테스트 환경을 지원하는 툴을 사용하여 원격 서버에 프로젝트가 배포되어 있어서 디버깅하기가 쉽지 않았는데
다양한 오류를 비교적 쉽게 해결할 수 있었다. 특히 API를 이용한 코드를 디버깅 및 테스트하면서 각 오류 상황에 따른
http요청을 서버에 전송하여, 정확히 어떤 헤더, 응답값들이 실려오는지 상세하게 테스트해 오류 해결 시간을 크게 단축시킬 수 있었다.

- GitGub

버전 제어 및 공동 작업을 위한 프로젝트 관리를 위해 사용하였다. 깃허브를 통해 팀원 긴 소스 공유를 하여
각자 개발상황을 원활히 알 수 있어서 편했고, 각 기능 간 취합이 용이하여 프로젝트가 수월하게 진척되었다.

- Putty : **KT 원격지 서버에 접속해 프로젝트를 배포**하여 실제로 사용자들이 카카오톡에서 챗봇 이용이 가능하도록 하였다.

리눅스 Ubuntu 서버로 명령어들을 사용하여 원격지 서버의 디렉토리를 파악할 수 있었다.

Git repository URL : https://github.com/JisooJang/KakaoRest-spring_boot

Programming Project Portfolio

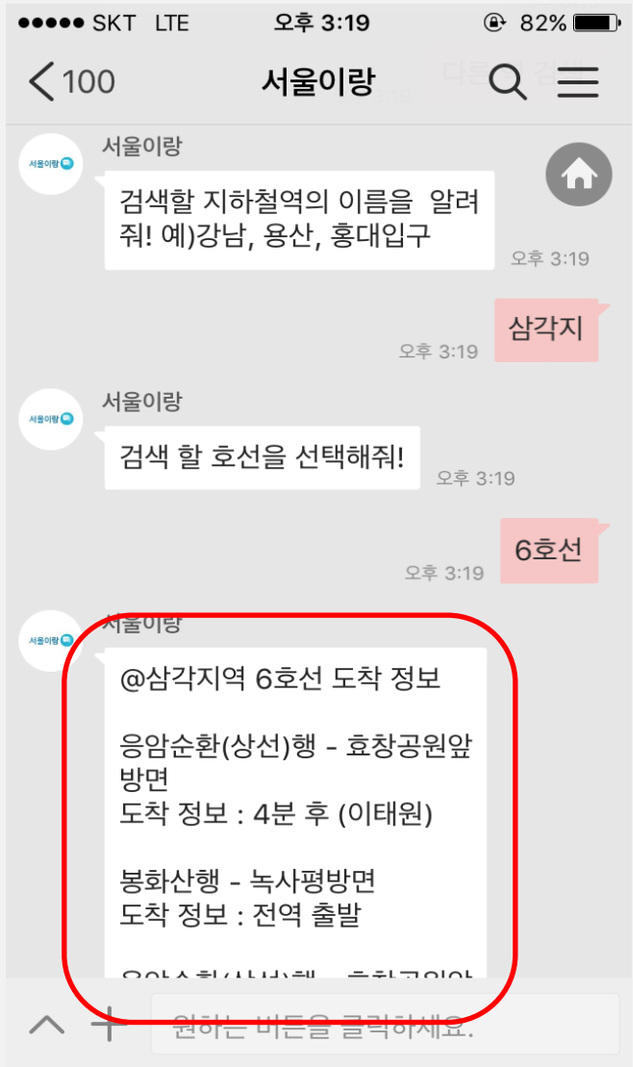
Chatbot-project '서울이랑'

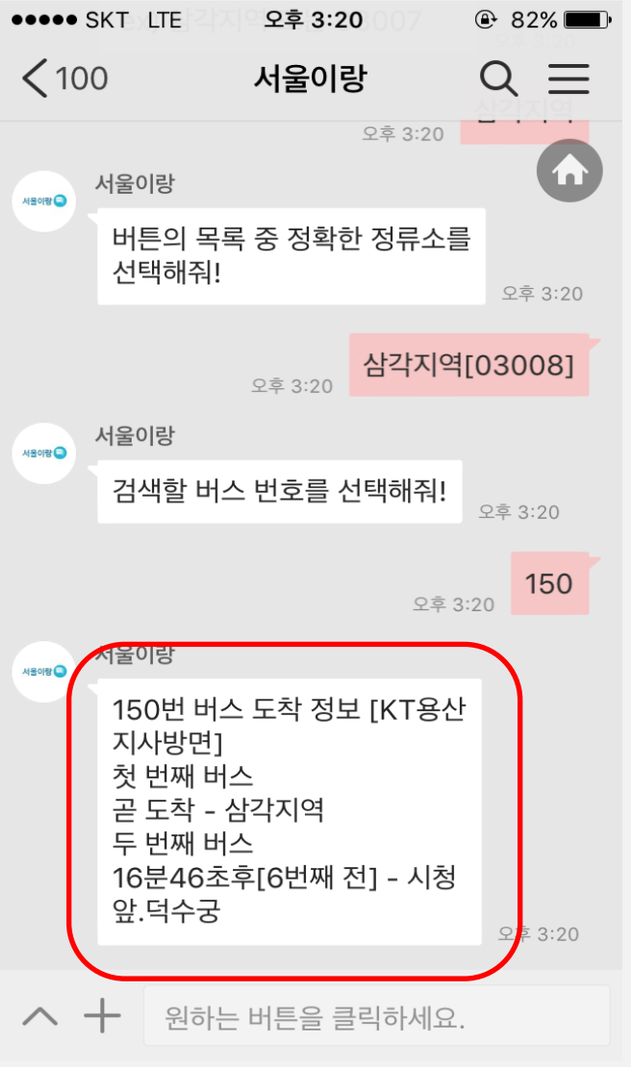
담당 기능

- 지하철 도착 정보, 버스 도착 정보 제공
- 최초 화면의 버튼 목록을 제공하는 메소드 정의
- kakaoMessage 클래스에서 **각 기능별 호출 클래스를 인터페이스 패턴**으로 분리

공공데이터 API를 이용하여 사용자가
지하철역명, 버스정류소명(또는 정류소번호) 입력
1차로 해당 **역 호선 정보와 정류소번호 목록**을 가져와
버튼 식으로 정확한 정보를 요구
버튼 클릭 시 2차로 해당 위치나 장소에 대한
교통 정보를 실시간 제공

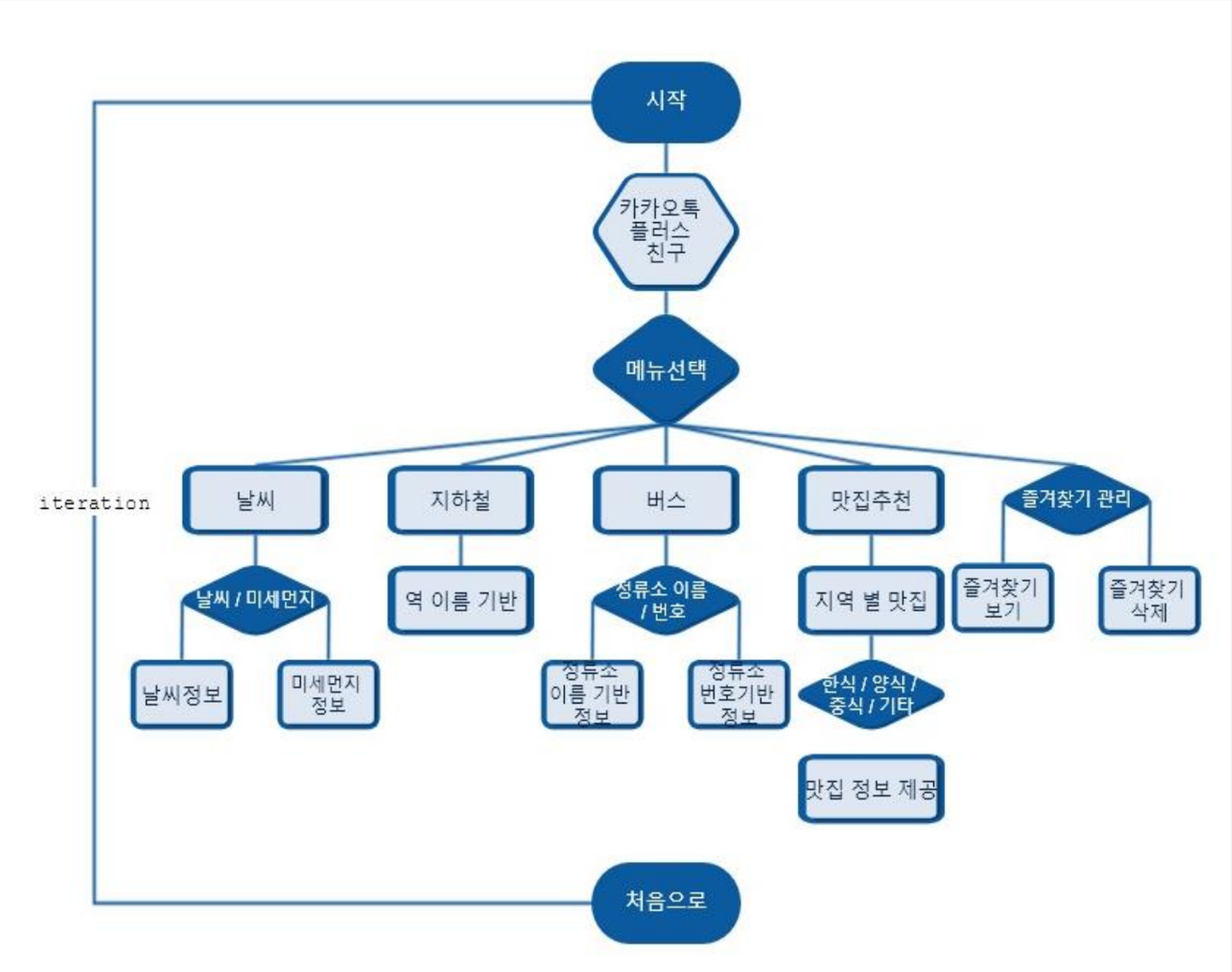
지하철의 경우 **양 방향 도착 정보**를 제공
버스의 경우 **2번째 버스 도착 정보**까지 제공





Programming Project Portfolio

Chatbot-project '서울이랑' - 전체 기능 프로세스



Programming Project Portfolio

Chatbot-project '서울이랑' - 메뉴 구성도

| 메뉴 구성도

1) 최초 이용자 화면 (자동응답 명령어 목록 호출) - 시스템 구성도

사용자 요청
Method : GET
URL : http(s)://:server_url/keyboard
Content-Type : application/json; charset=utf-8

사용자

MainController

MainMenu domain

String weatherInfo; // 날씨정보
String trafficInfo; // 교통정보
String customized; //개인 알람, 추천

Keyboard initMenu()

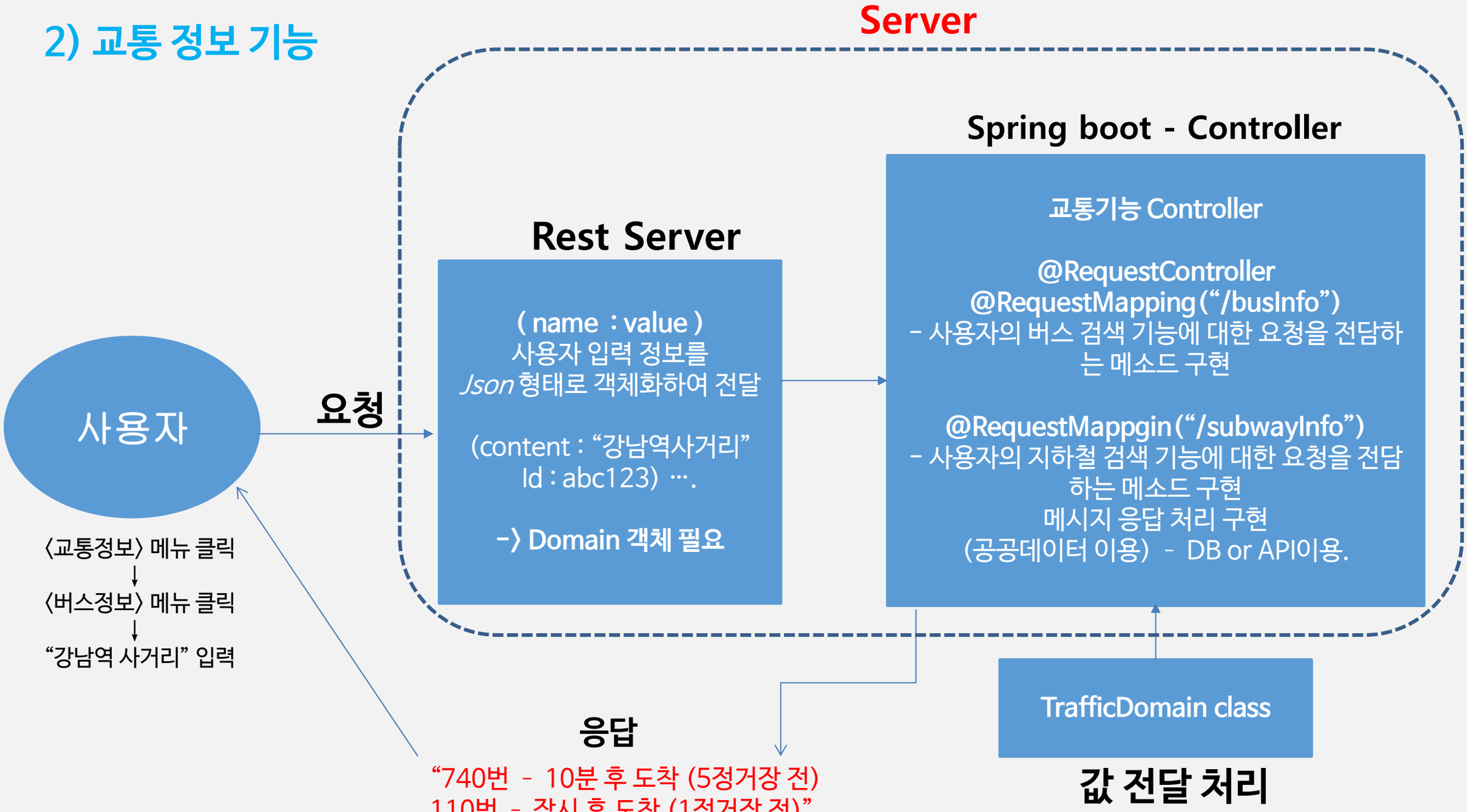
MainService

getMenuList()
InsertMenu()
deleteMenu()
modifyMenu()
……

Response

Return Keyboard;
{ "type" : "buttons", "buttons" : [“날씨 정보", “교통 정보", " 개인별 알람 및 추천 설정"] }

2) 교통 정보 기능


Server
Spring boot - Controller
교통기능 Controller
@RequestController
@RequestMapping(“/busInfo”)
- 사용자의 버스 검색 기능에 대한 요청을 전담하는 메소드 구현
@RequestMappgin(“/subwayInfo”)
- 사용자의 지하철 검색 기능에 대한 요청을 전담하는 메소드 구현
메시지 응답 처리 구현
(공공데이터 이용) - DB or API이용.
Rest Server
(name : value)
사용자 입력 정보를
Json 형태로 객체화하여 전달
(content : “강남역사거리”
Id : abc123) ….
-〉 Domain 객체 필요
사용자
요청
〈교통정보〉 메뉴 클릭
〈버스정보〉 메뉴 클릭
“강남역 사거리” 입력
응답
“740번 - 10분 후 도착 (5정거장 전)
110번 - 잠시 후 도착 (1정거장 전)”
TrafficDomain class
값 전달 처리

2) 교통 정보 기능 - 시스템 구성도

사용자 request
Method : POST
URL : http(s)://:your_server_url/message
Content-Type : application/json; charset=utf-8

curl -XPOST 'https://:your_server_url/message' -d
'{ "user_key": "1234",
"type": "text",
"content": " 강남역사거리" }'

사용자

"강남역 사거리" 입력

TrafficInfoController

TrafficInfo domain

busInfo(String user_key, String content)
subwayInfo(String user_key, String content)

TrafficInfoService

Dao.selectTrafficInfo()

TrafficInfoDao

select

TrafficInfo - 공공데이터 API

API

Response

{ "Message":{ "text" : "강남역 사거리 도착 정보 : ……"} }
"Keyboard": { "type": "buttons", "buttons": ["처음으로", "다른 검색", " 즐겨찾는 구간 추가"] } }

(Member DB에 추가)

Programming Project Portfolio

Chatbot-project ‘서울이랑’

각 기능별 클래스 패턴화

사용자의 모든 요청 -〉 **KakaoMessage** 클래스로 전달

어떤 기능을 요청했는지에 상관없이 **사용자의 모든 요청이 일단 KakaoMessage 객체로만 전달되는 구조**여서
정적 변수를 사용하여 사용자가 각 기능에서 몇 번째 프로세스에 있는지 체크하여,
각 프로세스에 맞는 메소드를 제공
사용자가 어떤 기능을 요청 중인지 기억하는 정적 변수에 따라,
각 기능별로 패턴화된 인터페이스를 구현한 클래스를 만들어 메소드 호출

적절한 정적 변수 사용과
패턴화된 **인터페이스**를 구현한 기능별 클래스들을 구조화
각 기능들이 서로 엉켜서 엉뚱한 기능이 호출되었던 문제들을 해결 가능

다수가 코드 협업 시 발생할 수 있는 문제점들을 해결한 부분이라 배운 점이 많았다.

Programming Project Portfolio

Chatbot-project '서울이랑'

챗봇 메인 버튼

지하철 | 버스 | 맛집추천 | 날씨 | 미세먼지 | 즐겨찾기 | 처음으로

각 버튼 요소는 List<String>에 저장 --- button_list

Map<String, Object> 구조의 HashMap에 ("buttons", button_list)로 저장 --- keyboard_map

JsonObject.put("keyboard", keyboard_map)을 추가한 후

jsonObject.toString()을 return하여 kakao API의 버튼 제공 구조에 맞춤

Programming Project Portfolio

Chatbot-project ‘서울이랑’

메인에서 지하철 버튼 클릭

지하철

도착정보는 지하철역 도착정보 API를 이용하여
호선과 지하철역을 설정하여 정보를 얻어와서 제공

1. 지하철 도착 정보를 알고싶으면 검색할 지하철역의 이름을 알려줘!
예)강남, 용산, 홍대입구

2. 지하철 역 간 최단경로 및 시간을 알고싶으면 출발역과 도착역을 함께 알려줘! (공백 필수)
예)삼각지 홍대입구

“삼각지” 입력

검색할 호선을 선택해줘!

4호선

6호선

“6호선” 클릭

호선버튼은 지하철역 정보 API를 이용하여
관련 호선 정보를 얻어와서 버튼으로 뿌려줌
**단, 호선이 한 개인 경우는 이 단계를 건너뛰고
바로 도착정보를 제공**

@삼각지역 6호선 도착 정보

응암순환(상선)행 - 효창공원앞 방면
도착 정보 : 4분 후 (이태원)

봉화산행 - 녹사평방면
도착 정보 : 전역 출발

**양방향의 2번째
도착정보까지 알려줌**

〈Button〉

다른 역 검색

즐겨찾는 역 추가

처음으로

**각 버튼 클릭 시
해당 프로세스로
돌아가도록 설계**

Programming Project Portfolio

Chatbot-project ‘서울이랑’

메인에서 지하철 버튼 클릭 지하철

공공데이터 rest API 이용

RealtimeStationArrival response = restTemplate.getForObject(url, RealtimeStationArrival.class, api_key, service_name, station_name)

url 형식 : http://swopenAPI.seoul.go.kr/api/subway/**{api_key}**/json/**{service_name}**/0/20/**{station_name}**

RealtimeStationArrival : rest API 응답값 형식에 맞게 작성한 java 클래스

Service_name에 따라 API 서비스를 변경하여 요청할 수 있음

Programming Project Portfolio

Chatbot-project '서울이랑'

지하철 정보 검색 기능 구현 프로세스

1) 사용자가 메인에서 **'지하철'** 버튼 클릭
2) 사용자는 **검색할 지하철 역 명**을 입력
3) 입력값이 올바른 역 이름이면, **Spring의 restTemplate 메소드**를 이용하여

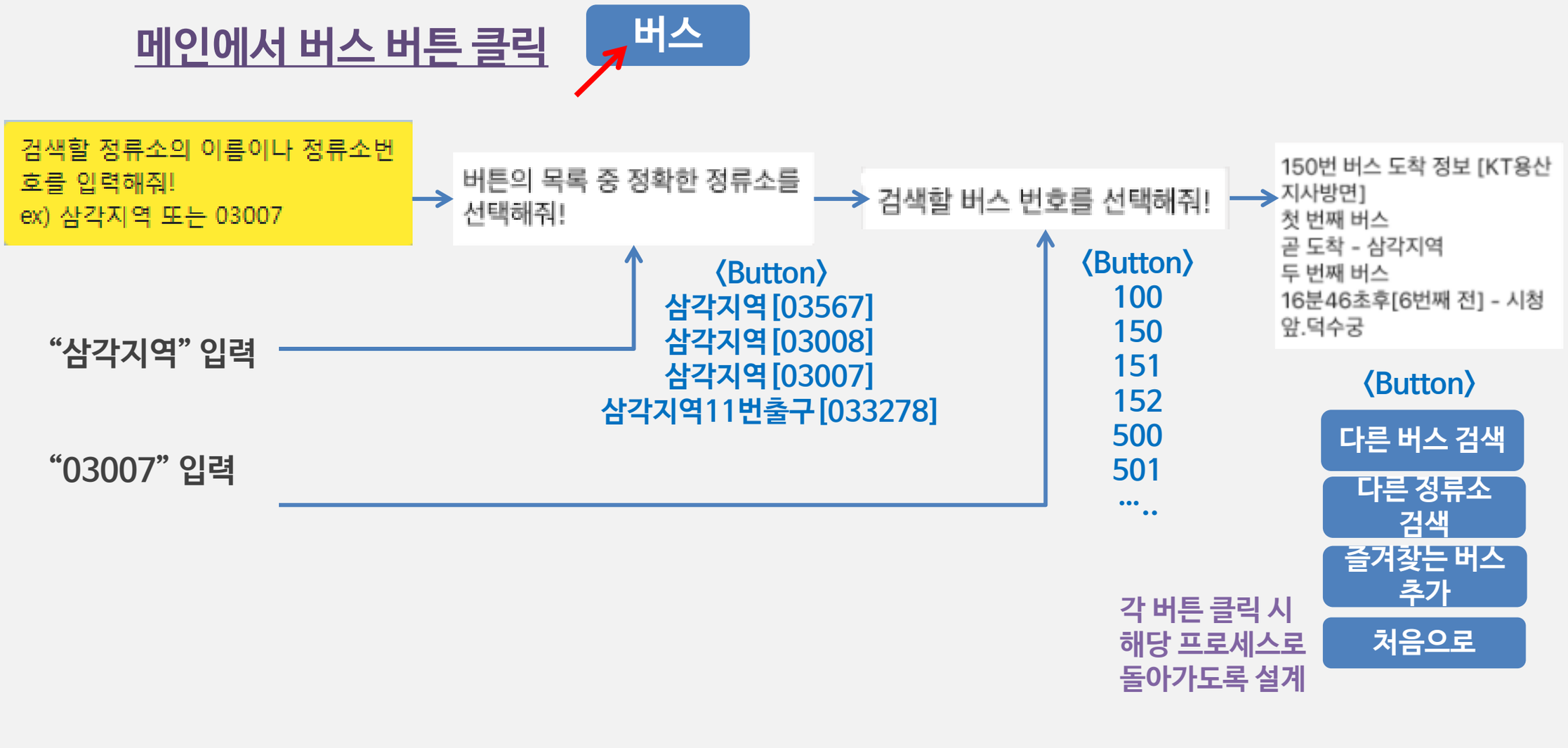
공공데이터 지하철 역 정보rest API를 활용해 **get 요청 URL에 정류소번호를 붙여서 요청**
응답값 형식에 맞게 미리 작성해놓은 클래스로 저장하여 재활용 가능

4) 먼저 입력된 지하철 역 명의 호선 정보를 불러와, **호선이 여러 개인 경우**, 버튼으로 **특정 호선을 선택**하게 함
(단 호선이 한 개인경우, 바로 역 도착정보를 제공)
5) 특정 호선을 선택하면, 사용자가 입력한 **지하철역명, 선택한 호선 정보**를 **요청 URL에 추가**하여

지하철역 도착 정보 rest API를 활용하여 get 요청 후 데이터를 받아와 사용자에게 도착정보를 제공
응답값 형식에 맞게 미리 작성해놓은 클래스로 저장하여 재활용 가능

Programming Project Portfolio

Chatbot-project '서울이랑'



Programming Project Portfolio

Chatbot-project '서울이랑'

메인에서 버스 버튼 클릭

공공데이터 rest API 이용

GetStationByUidItem response = *restTemplate.getForObject(url, GetStationByUidItem.class,*

operation_name, getDecoded_serviceKey(), station_num);

url 형식 : http://ws.bus.go.kr/api/rest/arrive/**{operation_name}**?serviceKey=**{service_key}**&stSrch=**{station_name}**

RealtimeStationArrival : rest API 응답값 형식에 맞게 작성한 java 클래스

operation_name에 따라 API 서비스를 변경하여 요청할 수 있음

Programming Project Portfolio

Chatbot-project ‘서울이랑’

버스 정보 검색 기능 구현 프로세스

1) 사용자가 메인에서 **‘버스’** 버튼 클릭
2) 사용자는 **정류소 이름이나, 정류소 번호**를 입력
3) 입력값이 정류소 번호이면, **Spring의 restTemplate 메소드**를 이용하여
공공데이터 rest API를 활용해 get 요청 URL에 정류소번호를 붙여서 요청

4) 입력값이 정류소 이름이면, 공공데이터 정류소 이름 목록 API를 활용해 **해당 문자열을 포함한 정류소 목록**을 모두 가져와 사용자의 버튼에 리스트로 제공

5) 위 3,4단계에서 정확한 정류소를 선택하면, **해당 정류소에 도착 예정인 모든 버스번호**를 버튼리스트로 제공
6) 정확한 버스버튼 클릭 시, 해당 **버스 정류소번호, 버스번호**를 이용해
공공데이터 버스 도착 정보 rest API를 활용하여 버스 도착정보 데이터를 받아와 사용자에게
도착정보를 제공