Google Calendar API를 이용한 Simple Calendar 개발과정

1. 3/21 ~ 3/23

1) 월뷰 캘린더를 그리고 그 위에 일정 표시. (하루만 가능)

2) 구조

* Google Calendar API에서 받아온 Event(일정 정보)를 CalendarDTO에 저장.
  + CalendarDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| title | String |  | 일정의 제목 | Event.getSummary() |
| start | String |  | 일정의 시작 시간 | Event.getStart().getDateTime() |
| startYMD | int | 3 | 일정의 시작 시간을 년, 월, 일로 기록 | Start의 년, 월, 일을 int로. |

* CalendarDTO에서 title과 startYMD[2](날짜 중 일 데이터만) 각각 String 배열, int 배열 형태로 View에 데이터 전송
  + 배열의 순서는 이벤트들을 API에서 받아온 순서(이벤트 시작 날짜 순)
  + 이 순서를 키 값으로 사용.
  + View에서는 각각을 eventSummaryList와 eventDateList 배열에 저장.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| eventSummaryList | String | 일정 개수 | 일정의 제목 | CalendarDTO.title |
| eventDateList | int | 일정 개수 | 일정의 시작 날짜 중 일 | CalendarDTO.startYMD[2] |

* 데이터를 받아서 달력을 그릴 때
  + 그리는 날짜 == eventDateList[key] 으로 맞는 날짜를 찾아감.
    - 키 값을 증가 시켜 모든 이벤트들을 달력에서 보이도록 함.
    - 같은 날에 여러 일정이 있을 수 있으므로 위 경우에서 키 값을 증가시켜 날짜 != eventDateList[key] 일 때 까지 추가.
* JSP
  + Index(월 뷰 캘린더)
  + MonthlyCalendar(월 뷰 캘린더)
* Class
  + HomeController
    - View에서 년도, 월을 받아서 그 날짜에 맞는 일정들을 GoogleCalendarService를 통해 가져옴.
    - 가져온 일정들을 View에 전달.
  + GoogleCalendarService
    - Google Calendar API를 이용하기 위해 인증을 함.
    - HomeController의 요청에 따라 일정들을 CalendarDTO 형태로 저장.
  + CalendarDTO

3) 문제점

* 각각을 배열들로 전달할 경우 비효율적일 수 있다.
* 지금 프로그램 구조로는 일정 중 여러 날에 걸친 일정은 표시할 수 없다.
* GET 방식으로 전달하고 있어 속도가 느리다.
  + AJAX 사용
* 지금의 DTO로는 일정에 대한 정보가 부족하다.
  + 사용자가 상세한 일정을 보기 위해서는 정보가 더 필요.
* 사용자가 여러 캘린더를 가지고 있을 경우를 고려하지 않았다.

2. 3/23 ~ 예상

1) 사용자가 가지고 있는 여러 캘린더의 일정들을 월 뷰 캘린더에 보여준다.(이어지는 일정도 가능)

2) 구조

* 구조 1안
  + CalendarDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| calendarID | String |  | 사용자의 캘린더 아이디 | CalendarListEntry.getID() |
| calendarSummary | String |  | 사용자 캘린더 제목 | CalendarListEntry.getSummary() |
| event | Event |  | 일정 | Service.events().list(calendarID).getItems().get() |

* + 캘린더 마다 따로 ArrayList
  + 이 데이터를 View로 전송.
  + 이벤트 추가, 수정 시 사용.
  + 장점
    - 일정 상세보기 요청이 쉬워 짐
    - 코딩, 구조가 간단함.
  + 단점
    - 메모리 차지하는 크기가 커짐.
    - 데이터 전송 시 느려 질 수 있음.
* 구조 2안
  + CalendarDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| calendarID | String |  | 사용자의 캘린더 아이디 | CalendarListEntry.getID() |
| calendarSummary | String |  | 사용자 캘린더 제목 | CalendarListEntry.getSummary() |
| eventID | String |  | 일정 아이디 | Event.getId() |
| summary | String |  | 일정 제목 | Event.getSummary() |
| start | DateTime |  | 일정 시작 날짜 | Event.getStart().getDateTime() |
| startTime | Int | 5 | 일정 시작 날짜를 년, 월, 일, 시, 분으로 나눔. | start를 년, 월, 일, 시, 분으로 나눔. |
| end | DateTime |  | 일정 끝 날짜 | Event.getEnd().getDateTime() |
| endTime | Int | 5 | 일정 끝 날짜를 년, 월, 일, 시, 분으로 나눔. | end를 년, 월, 일, 시, 분으로 나눔. |
| description | String |  | 일정의 내용 | Event.getDescription() |
| location | String |  | 일정의 장소 | Event.getLocation() |

* + 캘린더마다 따로 ArrayList
  + 이 데이터를 view로 전송
  + 이벤트 상세 정보를 보거나 수정, 삭제 시
  + 장점
    - 1안으로 바꾸기 쉬움
    - 전송이 빠름
    - 메모리 차지하는 크기가 적음.
  + 단점
    - 이벤트 상세 정보를 보려면 API요청을 해서 다시 Event를 얻어야 함.
    - 이벤트를 받아서 데이터 처리를 해야 함.
* 구조 2안으로 결정.
* JSP
  + Index(월 뷰 캘린더)
  + 월, 주간, 일, 목록 캘린더
  + 일정 상세정보
* Class
  + HomeController
    - View에서 년도, 월을 받아서 그 날짜에 맞는 일정들을 GoogleCalendarService를 통해 가져옴.
    - 가져온 일정들을 View에 전달.
  + GoogleCalendarService
    - Google Calendar API를 이용하기 위해 인증을 함.
    - HomeController의 요청에 따라 일정들을 CalendarDTO 형태로 우선순위에 맞게 저장.
    - 우선순위를 정하기 위해 EventProcessing 클래스의 함수를 이용.
  + EventController
    - 일정 수정, 삭제를 담당하는 컨트롤러.
  + EventProcessing
    - 받아온 일정들을 우선순위에 맞게 정렬.
  + CalendarDTO

4)우선순위

* 일정이 여러 날에 거쳐서 나타날 경우
  + End Date가 늦은 순서 > 시간이 있는 일정 > 종일 일정 ( End Date가 같은 날 일 때도)
* 하루 일정 중 같은 날
  + 종일 일정 > 시간이 있는 일정.