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 순서(이벤트 시작 날짜 순)를 key 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 ~

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

2) 구조

* 구조 1안
  + EventDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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
  + EventDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| calendarID | String |  | 사용자의 캘린더 아이디 | CalendarListEntry.getID() |
| eventID | String |  | 일정 아이디 | Event.getId() |
| summary | String |  | 일정 제목 | Event.getSummary() |
| start | long |  | 일정 시작 날짜 | Event.getStart().getDateTime().getValue() |
| startTime | Int | 5 | 일정 시작 날짜를 년, 월, 일, 시, 분으로 나눔. | start를 년, 월, 일, 시, 분으로 나눔. |
| end | long |  | 일정 끝 날짜 | Event.getEnd().getDateTime().getValue() 여기서 종일 일정인 경우는 하루 빼줘야 함. |
| endTime | Int | 5 | 일정 끝 날짜를 년, 월, 일, 시, 분으로 나눔. | end를 년, 월, 일, 시, 분으로 나눔. |

* + 선택된 캘린더들의 일정은 같은 ArrayList에 들어가 있음.
  + 이 데이터를 view로 전송
  + 이벤트 상세 정보를 보거나 수정, 삭제 시
  + +) 이벤트가 많아지면서 전송 데이터 사이즈가 커져서 start, end는 long으로, description, location은 캘린더에서는 보지 않기 때문에 제외.(일정을 클릭 해야 볼 수 있음)
  + 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.
* DTO
  + CalendarDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| id | String |  | 사용자의 캘린더 아이디 | CalendarListEntry.getId() |
| summary | String |  | 사용자의 캘린더 제목 | CalendarListEntry.getSummary() |
| check | boolean |  | 사용자가 캘린더의 체크박스 체크 여부 | Calendar의 id를 name으로 갖는 세션의 value  (초기는 true) |
| colorId | String |  | 사용자의 캘린더 색 아이디 | CalendarListEntry.getColorId() |
| primary | Boolean |  | 캘린더가 사용자의 메인 캘린더인지에 대한 여부 | calendarListEntry.getPrimary() |
| accessRole | String |  | 캘린더의 Access Role(일정에 대한 접근 권한 정의) | calendarListEntry.getAccessRole() |

* + CheckedCalendarDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| id | String |  | 체크 여부가 바뀐 캘린더 Id | View에서 전달 |
| year | int |  | 사용자가 보고 있는 년도 |  |
| month | int |  | 사용자가 보고 있는 월 |  |
| date | int |  | 사용자가 보고 있는 일 |  |

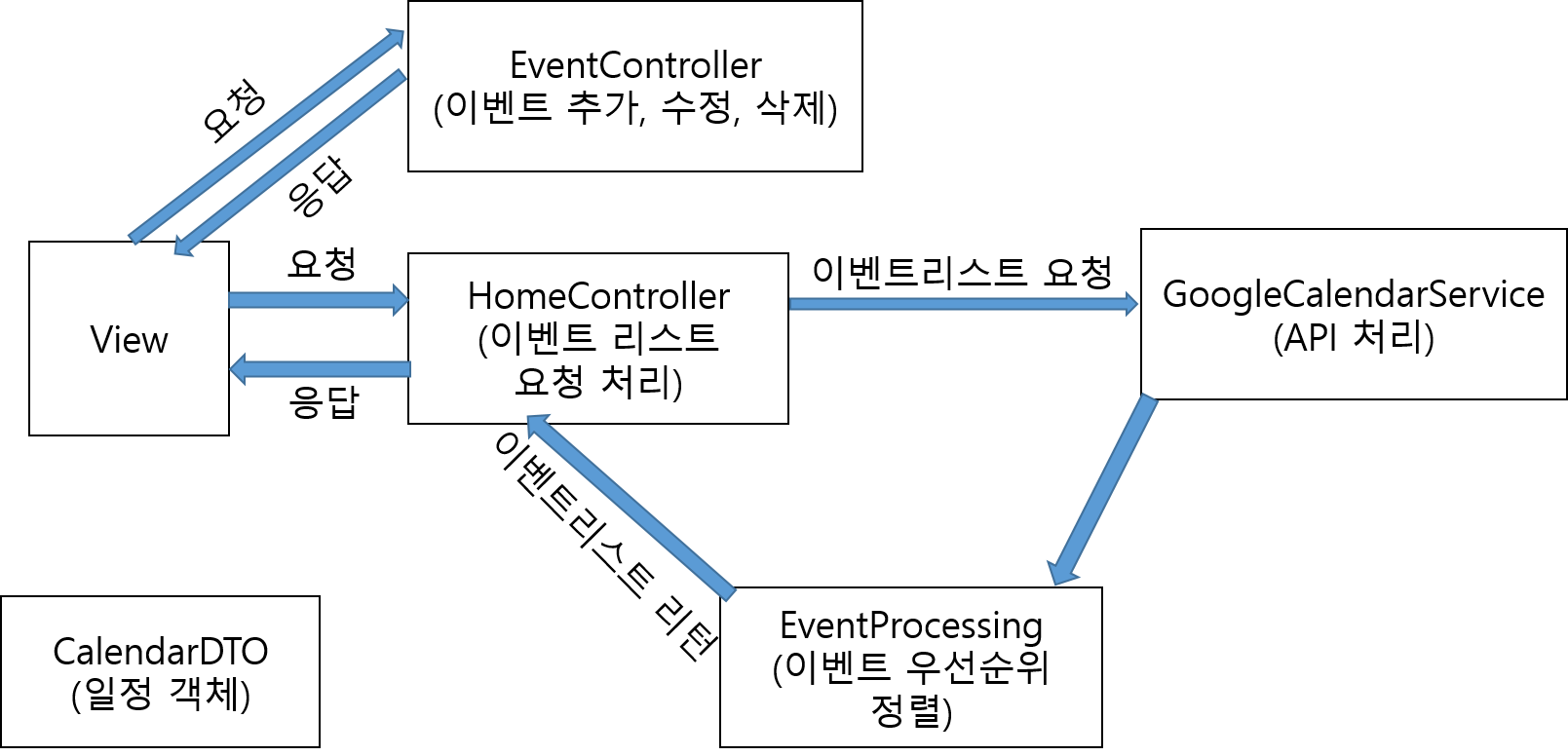
* + - 캘린더 체크 여부에 따라 view에서 controller로 요청을 보낼 때 어떤 캘린더가 체크 여부가 바뀐 지, 현재 사용자가 보고 있는 날짜는 언제인지에 대한 데이터를 가지고 있음.
  + EventDetailDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| summary | String |  | 일정 제목 | Event.getSummary() |
| startTime | Int | 5 | 일정 시작 날짜를 년, 월, 일, 시, 분으로 나눔. | 시작 날짜를 년, 월, 일, 시, 분으로 나눔. |
| endTime | Int | 5 | 일정 끝 날짜를 년, 월, 일, 시, 분으로 나눔. | 끝 날짜를 년, 월, 일, 시, 분으로 나눔. |
| description | String |  | 일정 상세 내용 | Event.getDescription() |
| Location | String |  | 일정 장소 | Event.getLocation() |
| Recurrence | List<String> |  | 일정 반복 규칙 | Event.getRecurrence() |

* + - 달력에 있는 일정의 제목을 눌러서 요청이 들어왔을 경우 보내주는 데이터 구조
      * Google Calendar API에서 제공해주는 Event객체를 이용하려 했으나 데이터 사이즈 크기도 크고 시작, 끝 날짜를 View에서 표현하기 어려움.
  + CalendarAndEventIdDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| calendarId | String |  | 일정 정보 요청한 일정의 캘린더 Id | View에서 전달 |
| eventId | String |  | 일정 정보 요청한 일정의 Id |  |

* + - View에서 일정 제목을 클릭해서 요청을 할 경우 일정의 캘린더 Id와 이벤트 Id를 이 구조 형태로 담아 보냄.
    - Controller에서는 이 두개의 아이디를 이용해 일정의 정보를 Google Calendar API로 요청해 정보를 얻음.
* JSP
  + Index(캘린더)
    - 현재 사용자가 보고 있는 날짜를 YYYY-MM-DD형태로 url에 추가.
      * /m/YYYY-MM-DD 형태
    - 이것을 이용해서 사용자가 보고 있는 날짜 추출.
    - 달력을 그린 뒤 이벤트들을 달력에 그림.
    - 사이드에 사용자의 캘린더 리스트를 체크박스로 보여줌.
      * 클릭에 따라 체크된 캘린더에 해당하는 일정만 보여줌.
    - 일정의 제목을 누르면 일정에 관한 간단한 정보를 보여줌.
  + 일정 상세정보(여기에서 추가, 수정, 삭제 가능하게)
    - Google Calendar API의 Event JSON 데이터를 받음.
* Class
  + HomeController
    - GoogleCalendarService에서 캘린더 리스트 받아서 초기 화면에 전송.
    - View에서 년도, 월을 받고 사용자가 선택한 캘린더에 있고 그 날짜에 맞는 일정들을 GoogleCalendarService를 통해 가져옴.
    - 가져온 일정들을 View에 전달.
    - 캘린더 선택이 달라질 때 마다 재요청.
  + CalendarController
    - 캘린더 목록 요청이 들어오면 GoogleCalendarService를 통해 캘린더 목록을 받아 리턴.
    - 캘린더 체크에 따라 달라짐.
    - Session에 캘린더의 체크 여부를 저장. name = CalendarId, value = 체크 여부
      * 이를 통해 체크 된 캘린더 리스트를 리턴
  + GoogleCalendarService
    - Google Calendar API를 이용하기 위해 인증을 함.
    - HomeController에 캘린더 리스트 보내주기.
    - HomeController의 요청에 따라(날짜, 캘린더) 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.
    - 시작 날짜가 현재 요청한 월 보다 전인 이벤트들을 모아서 이벤트 끝 날짜가 늦은 순서로 정렬
    - 그 외에는 시작 날짜가 같은 날인 이벤트들을 모아서 이벤트 끝 날짜가 늦은 순서로 정렬.
  + EventDTO
    - Google Calendar API로부터 받은 이벤트 정보를 필요한 정보만 뽑아 저장.
  + CalculateCalendar
    - 해당 월의 마지막 날을 계산해서 리턴 해주는 함수가 있음.
  + CalendarDTO
    - Google Calendar API로부터 받은 사용자의 캘린더들에 대한 정보를 필요로 하는 것만 뽑아서 저장.
  + EventDetailDTO
    - View에서 사용자가 일정 제목을 누르면 보여줄 정보를 담는 데이터 구조.
  + CalendarAndEventIdDTO
    - View에서 사용자가 누른 일정 제목의 Calendar ID와 Event ID를 담는 데이터 구조.
  + CheckedCalendarDTO
    - 사용자가 클릭한 체크박스의 캘린더 ID와 현재 사용자가 보고있는 년도, 월, 일을 담는 데이터
      * 이를 통해 체크박스 여부를 업데이트 한 후 사용자가 보고있는 날짜에 대한 일정 정보를 View에 보내줌.



* 추후 추가 예정. 컨트롤러 클래스의 분리가 필요.

4)우선순위

* 일정이 여러 날에 거쳐서 나타날 경우
  + End Date가 늦은 순서
* 시작 날짜와 끝 날짜가 같은 일정인 경우
  + 종일 일정 > 시간이 있는 일정.
  + 시간이 있는 일정인 경우는 시작 시간이 빠른 순으로 정렬.
* 위 순서에 맞게 전체 일정 정렬 후 이벤트 리스트를 리턴.
* View에서는 순서대로 처리.
* 현재 보고 있는 월보다 시작 날짜가 전인 경우
  + 끝 날짜가 늦은 순서
* 그 외의 경우
  + 시작 날짜와 끝 날짜가 같은 일정인 경우
    - 종일 일정 > 시간이 있는 일정 순서
    - 시간이 있는 일정인 경우에는 시작 시간이 빠른 순으로 정렬.

5)예외처리

* 월, 일 범위 검사해주는 함수 필요.

\*\* Start Time은 inclusive고 End Time은 Exclusive

그래서 종일 이벤트인 경우는 End Time에 하루를 더 더해줘야 함.

6)기능

* 일정 추가의 경우
  + 현재 보고 있는 달의 첫 번째 날을 넣어줄지
  + 현재 날짜를 넣어줄지
* 일정 입력 시
  + 시작 날짜가 끝 날짜 보다 뒤면 둘 다 시작 날짜에 맞추기
  + 끝 날짜가 시작 날짜 보다 먼저면 시작 날짜에 맞추기
  + 시간도 마찬가지로.
    - 종일 일정을 누르지 않고 시간의 X버튼을 누를 경우 0시 0분으로 세팅 됨.
    - 이 경우 끝 시간은 시작 시간보다 1시간 뒤.
      * 11시인 경우는 30분 전까지는 30분을 더하고 그 후는 더하지 않는다.
* 더보기
  + 스크린 위치 따라 위치 조정