진행 정도 기록

0. 그 전

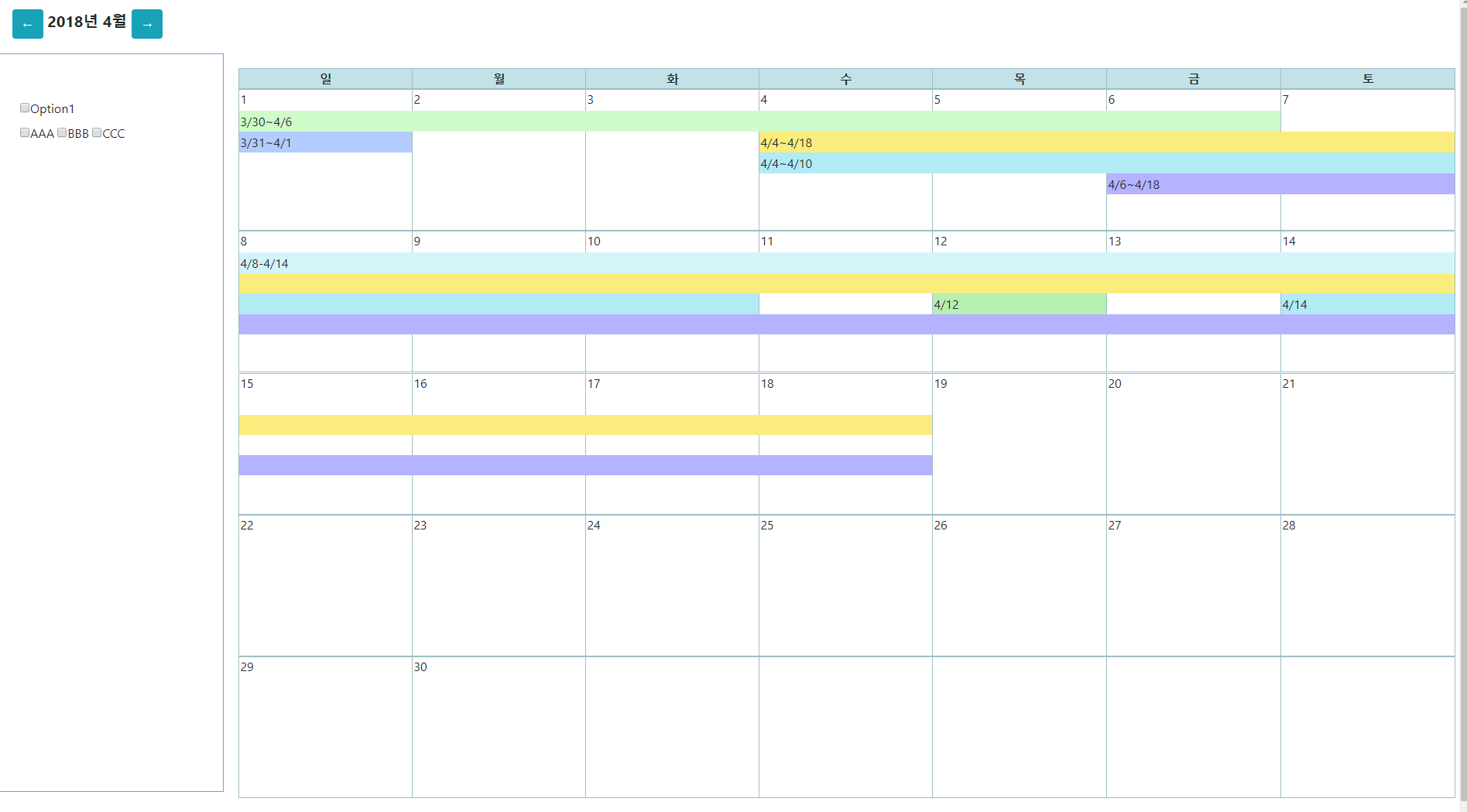
* Javascript를 이용하여 해당 날짜의 월 뷰 캘린더를 그림.
  + 오늘 날짜인 칸은 다른 색으로 표시.
  + 일정은 받아온 순서대로(시작 날짜 순) 표시.
* 그 전과 그 후 월을 보기 위해 back, forward 버튼 추가.
* GET, POST 방식.
  + 다른 월을 보려면 페이지를 계속 다시 로드 함.
* 일정의 시작 날짜, 제목을 각각 배열로 View에 전달.

1. ~ 2018-03-27

* View와 Controller 사이 전송 정리.
  + 초기 화면 index.jsp 요청
    - index.jsp 리턴
  + /{date} 요청
    - date는 YYYY-MM-DD 형태. 여기서 날짜를 추출해서 HomeController 클래스에 저장.
    - Index.jsp 리턴
  + 새로 고침을 할 때 마다(페이지 로드 될 때마다) /MonthlyCaledar 요청.
    - HomeController 클래스 내에 저장하고 있는 year, month에 맞는 일정 리스트 리턴.
  + Back 버튼, forward 버튼을 누를 시 /monthly/{year}/{month}/{date} 요청
    - year, month, date는 요청하는 날짜.
    - HomeController 클래스 내에 저장하고 있는 year, month 값 변경.
    - 그에 맞는 일정 리스트 리턴
    - url을 /YYYY-MM-DD(요청한 날짜)로 변경.
* AJAX 비동기 전송 방식으로 변경.
* 달력을 기존에는 표로 날짜를 표시하고 일정이 들어가면 해당 칸에 표를 추가하는 방식이었음.
  + 지금은 div로 요일, 날짜 영역을 나누고 날짜 영역을 한 주씩 div로 나눈 후 그 주 안에서 표를 그림. 7\*8 표로 그림. 맨 첫 줄은 각 칸에 해당하는 날짜가 들어가고 그 아랫줄부터 일정이 들어감.

2. ~2018.03.28

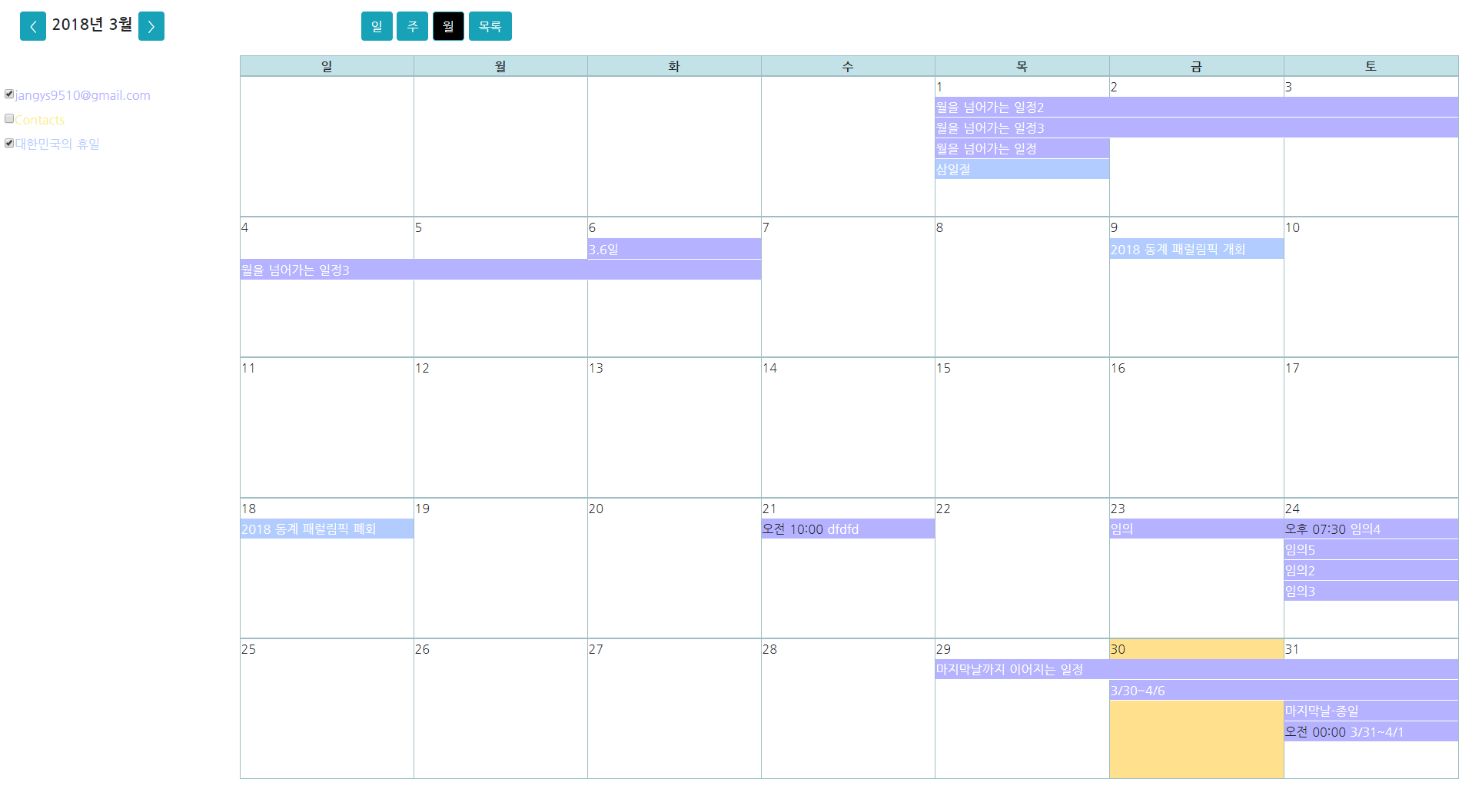
* 달력에 이어지는 일정 그리기
  + 주 안에 있는 표는 7\*7로 변경.
  + 일정이 들어가는 칸에는 0~(6+해당 월의 최대 주 개수\*7)-1를 값으로 순서대로 data-index 속성 부여.
    - 같은 주의 같은 열에는 같은 data-index가 들어감.
    - 같은 행에는 data-col이 들어감.(1~6)
  + 일정 그리기
    - 일정의 시작, 끝 날짜의 data-index 값 구하기
      * 일정의 시작, 끝 날짜의 data-index 값 계산법
        + 시작(끝) 날짜 + 해당 월의 시작 요일(0~6) -1
      * 시작 날짜가 해당 월보다 전이면 시작 날짜의 data-index는 해당 월의 첫번째 날의 data-index로 설정.
      * 끝 날짜가 해당 월보다 후이면 끝 날짜의 data-index는 해당 월의 마지막 날의 data-index로 설정.
    - 일정 그리기
      * 해당 주의 일정 날짜 끝과 시작 날짜 data-index 값을 가지는 첫번째 태그의 data-col속성 확인.
        + 둘 중 큰 값을 col에 저장.
        + Data-col과 data-index속성 값을 가지는 태그들을 찾아 설정.
    - 하루 일정인 경우
      * 일정의 시작 날짜의 data-index 값, data-col 값을 가지는 태그들 중 첫 번째 태그를 찾음.
      * 그 태그의 text, css 변경 후 data-index 속성 값 제거
        + 제거 하여 일정이 들어가지 않는 칸이 항상 첫 번째 태그가 되도록 함.
        + colspan을 이용해 일정 날짜 만큼 칸 합치고 합친 만큼 제거.
    - 이어지는 일정인 경우
      * 특정 주의 마지막 data-index는 6 + 7\*(주의 index(0~5))로 알 수 있음.
      * 이를 이용하여 시작 날짜 data-index가 어느 주에 포함되는지 계산.
      * 시작, 끝 날짜가 같은 주에 있는 경우, 주를 넘어가는 경우, 그 주의 첫번째 날~끝 날짜인 경우로 나눠서 구현
      * 시작 날짜 data-index와 가능한 data-col 값을 가지는 첫 번째 태그를 찾아 text, css 변경 후 속성 값 제거
      * 시작 날짜는 아니지만 그 주에서 첫 번째 날인 경우는 해당 data-index 값을 가지는 첫 번째 태그를 찾아 css 변경 후 속성 값 제거.
      * Colspan을 이용해 일정의 날짜 만큼 칸을 합치고 그 수만큼 칸을 삭제.
    - 해당 주의 일정 날짜 끝과 시작 날짜 data-index 값을 가지는 첫번째 태그의 data-col속성 확인.
      * 둘 중 큰 값을 col에 저장.
      * Data-col과 data-index, data-col값을 가지는 태그들을 찾아 설정.
    - 일정의 끝 날짜가 0시 0분인 경우 그 전 날을 끝 날짜로 해서 이어지는 일정을 그림.
* 일정들을 우선순위에 맞게 정렬
  + 일정의 시작 날짜가 해당 월보다 전인 이벤트들을 모음.
    - 끝 날짜가 늦은 순서대로 정렬
      * 날짜의 값을 밀리초 단위로 받기 때문에 이 값이 큰 순서대로 정렬.
      * 종일 일정에는 끝 날짜가 하루 추가 되기 때문에 86400000 만큼 뺌.
  + 그 외의 경우 일정의 시작 날짜가 같은 이벤트들을 모음.
    - 끝 날짜가 늦은 순서대로 정렬.

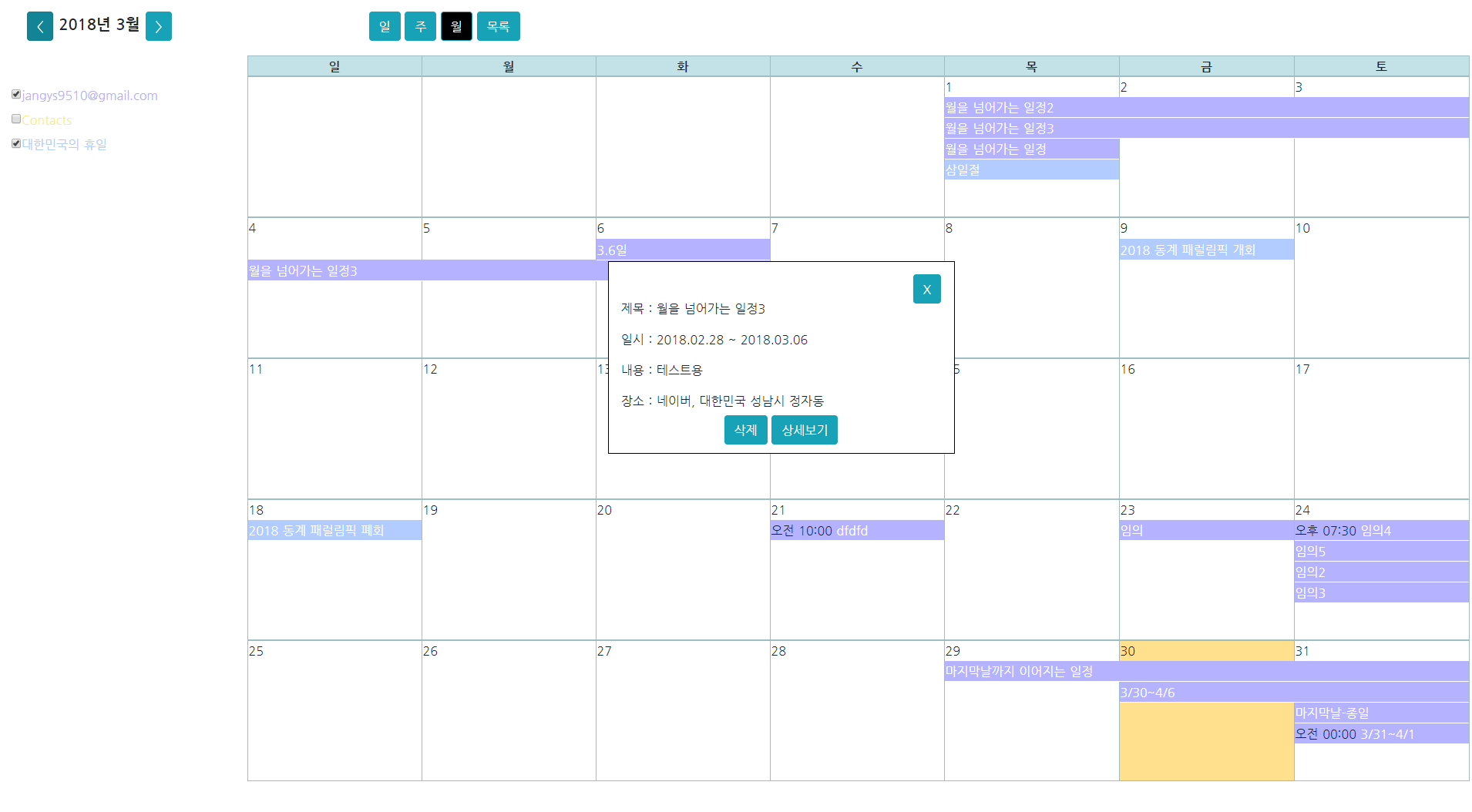


* 2018.03.28 캘린더 UI 모습
  + 월 뷰 캘린더. 일정을 보여줌. 다른 월도 확인 가능.

3. ~2018.03.30

* 일, 월, 주, 목록 버튼을 추가해 각 버튼을 클릭하면 달력 부분이 다른 것이 보이도록 함.
  + 확장성을 위해 요청, 달력 출력 부분을 함수로 분리
  + 각 버튼을 클릭할 때 마다 url을 변경해 추후 구분이 갈 수 있게 함.
    - d, m, w, l.
* 사용자의 여러 개의 캘린더를 체크박스 리스트에 보여줌.
  + 캘린더의 정보는 CalendarDTO 형태로 저장해서 View로 전달.
    - Id, summary, check, colorId
  + 페이지가 로드 되는 경우에 ajax로 사용자의 캘린더 리스트를 받음
  + 개수만큼 체크박스 생성
  + 캘린더의 colorId를 이용해 여러 색의 코드를 담은 배열에서 색을 얻음.
    - colorId는 정수 형태로 되어있는데 배열[colorId%배열사이즈]로 항상 특정 색을 얻어서 글자 색을 바꾸도록 함.
* Session에 캘린더의 체크 여부를 저장.
  + name = Calendar Id, value = 체크 여부 형태로 저장.
* 사용자가 어떤 캘린더를 선택했는지 컨트롤러에 전달해 조건에 맞는 이벤트를 보여줌.
  + 체크박스에 클릭 이벤트를 추가함
  + 캘린더의 id는 id@group.calendar.google.com이나 id@gmail.com(최초로 생성된 사용자의 캘린더)로 되어있음.
    - url로 요청할 때 id 값을 넣으면 뒤에 점이 있어서 온전한 값이 가지 않음.
    - 그래서 ajax요청시 json으로 데이터를 만듦.
    - CheckendCalendarDTO 형태. Id, year, month, date의 값을 가짐.
    - 이 데이터를 전송
    - Controller에서는 이 정보를 통해 Session에 저장된 값을 변경
    - Session에 저장된 캘린더들 중 체크 여부 값이 true인 Id만 리스트에 저장.
    - 위의 캘린더 Id에 해당하는 이벤트들을 받아 View로 전달.
* 캘린더 리스트를 요청할 때 마다 세션에 있는 캘린더 정보 갱신
  + 캘린더를 삭제한 경우가 있을 수 있으므로 현재 캘린더 리스트에 없는 Id값을 가지는 세션이 있다면 삭제.
* 일정을 캘린더 색에 맞게 배경색 지정
  + 일정의 캘린더 id와 같은 값을 가지는 체크 박스(value = calendarId)를 찾아 체크 박스의 color값을 가지고 배경색을 지정.
  + 일정이 표시되는 경우 border-bottom 값을 흰 색으로 해 다른 일정과 구분이 갈 수 있게 함.
* 일정에 시간을 표시하고 제목을 눌러 요약된 정보를 볼 수 있게 함.
  + 시작 날짜에 시간 정보가 있는 경우 표시
  + 제목을 누르면 태그에 속성으로 저장한 일정의 CalendarId, EventId를 가지고 요청해 이벤트의 상세 정보를 받음
    - 받은 정보를 화면에 보여줌.
  + 정보를 볼 수 있는 창은 div로 구현. 달력 위에 겹쳐서 보이게 함.
    - X를 누르면 창을 끌 수 있음.(display=none)
    - 삭제, 상세정보를 볼 수 있는 버튼이 있음(눌렀을 때 이벤트는 아직 없음)





* 현재까지의 UI.

4. ~2018.04.04

* 일정 삭제 기능 추가.
  + 삭제 버튼은 일정 제목을 누르면 나오는 요약 정보 창에 있음.
  + /deleteEvent URL로 요청하게 됨.
  + EventController에서 이벤트 삭제. Google Calendar API를 이용
    - Service.events().delete(calendarId, eventide).execute()
* 일정 상세보기 페이지 추가
  + 이 페이지에서 일정의 수정, 생성을 함.
  + 현재까지는 제목, 시작과 끝 날짜, 장소, 메모, 캘린더만 저장할 수 있음.
  + 위 정보를 같은 form에 넣어서 Controller에서 request.getParameter(“name”)으로 입력된 값을 가져오게 함.
    - /updateEvent URL로 Post 방식으로 요청. accept-characterset=”UTF-8”로 해야 한글이 깨지지 않는다.
    - 저장 버튼을 누르면 submit이 되고 취소 버튼을 누르면 history.back()으로 뒤로 가기가 됨.
  + 일정의 수정과 생성은 같은 함수에서 관리.
  + 수정을 할 때는 수정할 EventId, CalendarId 정보를 text input에 보이지 않게 저장.
    - 위 정보를 가지고 Event의 상세한 정보를 요청.
    - Google Calendar API의 Event객체로 보내줌.
    - 위 정보를 가지고 input 태그안에 값을 넣어서 보여줌.
    - 날짜 정보는 “yyyy-MM-dd”형태로 저장되고(valueAsDate) 시간 정보는 “hh:mm:ss”(value)로 저장 됨.
  + 생성을 할 때는 현재 날짜를 시작과 끝 날짜에 입력해 보여주고 종일 이벤트를 체크하지 않음. 시간 정보는 현재 시간을 시작 시간으로, 현재 시간에서 한시간 더한 것을 끝 시간으로 설정.
  + 날짜 유효성 검사 후 변경 기능.
    - 시작 날짜가 끝 날짜보다 늦은 날로 입력되거나 끝 날짜가 시작 날짜보다 빠른 날로 입력된 경우
      * 끝 날짜를 시작 날짜에 맞춤. 같은 날로 맞춰 줌.
      * 계산은 long value로 날짜가 늦거나 빠른 경우를 판단.
      * 끝 날짜가 종일인 경우는 하루가 추가된 경우이므로 86400000을 빼 줌.
    - 시간도 위의 날짜의 경우와 마찬가지.
      * 끝 시간을 시작 시간보다 한시간 뒤로 맞춤.
      * 시작 시간이 23시인 경우 23시 30분 전까지는 끝 시간을 23시 30분으로 맞추고 그 후는 시작 시간과 끝 시간을 같게 함.
      * 시간 계산은 시간 input의 값을 “:”을 기준으로 나눠 시, 분을 가지고 계산.
    - 날짜와 시간의 유효성은 onblur 이벤트로 처리.
  + 일정이 들어갈 캘린더 리스트를 보여줌.(select)
    - 수정의 경우 일정이 들어가 있는 캘린더를 먼저 선택해서 보여줌.
    - 모든 캘린더를 보여주는 것이 아님.
      * 캘린더의 Access Rule에 따라 캘린더의 일정을 수정, 생성 여부가 결정 됨.
      * Access rule이 writer나 owner인 경우만 가능.
      * 캘린더의 Access Rule이 writer나 owner인 캘린더만 리스트에 추가해서 보여 줌.
      * /CalendarList URL로 요청.
  + EventController에서 이벤트 생성, 수정에 대한 요청 처리
    - calendarId와 eventId는 보이지 않는 text input 태그에 저장.
      * 페이지를 불러올 때 Controller에서 model.addAttribute()로 두 값을 저장.
      * 이벤트를 생성하는 경우에는 두 값을 “addEvent”로 저장.
    - 입력된 정보를 모두 이용.
    - 날짜는 “-“를 기준으로 년도, 월, 일을 구함.
    - 시간은 “:”를 기준으로 시, 분을 구함.
    - 종일 일정인 경우(allDay != null)는 날짜 정보만을 가지고 Date 타입으로 만든 후(년도 -1900, 월 - 1) DateTime을 생성할 때 DateOnly를 true로 Date.getTime()을 통해 value값을 구하고 Date.getTimezoneOffset()로 값을 설정 함.
      * 이 값을 가지고 일정의 날짜를 세팅하는데 필요한 EventDateTime 형태로 다시 저장함. timeZone은 “Asia/Seoul”로 세팅.
      * 끝 날짜는 하루 뒤로 표현 해야 하므로(exclusive) value값에 86400000을 더해 줌.
    - 시간이 있는 일정인 경우에는 날짜 정보와 시간 정보를 가지고 종일 일정 처럼 Date->DateTime(Date값만 넣어도 됨)->EventDateTime으로 저장.
    - 이벤트 수정
      * 먼저 수정할 이벤트를 calendarId, eventId를 이용하여 구해 옴.
        + Service.events().get(calendarId,eventId).execute()
      * Event.setSummary()같은 함수들을 이용하여 변경된 내용 세팅.
        + Request.getParameter(“name”)으로 form에 입력된 값을 가져 옴.
      * Service.events().update(calendarId,eventide,event).execute()로 수정.
      * 사용자가 일정의 캘린더를 다른 캘린더로 바꿨을 경우
        + update에는 calendarId를 바꾸는 기능이 없음.
        + 일정을 지운 후 바뀐 calendarId로 다시 생성.
    - 이벤트 생성
      * 이벤트를 생성하는 경우에는 수정할 때 와 동일한 함수를 사용.
      * 날짜와 시간에 대한 처리는 위와 동일.
      * 생성은 사용자가 캘린더를 바꿔 수정할 때 다시 생성하는 부분과 동일하게 사용. calendarId가 선택된 calendarId와 다른 경우 임.
        + 이벤트를 생성하는 경우에는 caelndarId와 eventId값으로 addEvent를 넣어 두었으므로 선택된 calendarId와 같지 않음.
        + 새로운 이벤트 객체를 만들어 입력된 정보를 세팅함.

Service.events().insert(newCalendarId, event).execute()

* + - 완료 되면 redirect를 사용하여 m/이벤트의 시작 날짜 URL로 이동.
      * 이벤트의 시작 날짜에 해당하는 월 뷰 캘린더가 보임.
* 우선순위 정렬
  + 자바의 Comparator를 이용.
  + 여러 캘린더에서 이벤트를 받아오므로 초반 정렬이 필요함.
    - 시작 날짜가 빠른 순으로 정렬.(getStart(), value 값 이용.)
  + 정렬된 이벤트를 가지고 추후 이벤트를 그리기에 편하기 위해 재정렬.
    - 1) 현재 날짜보다 시작 날짜가 빠른 일정만 뽑아서 다른 리스트에 저장.
      * 이 리스트는 끝 날짜가 늦은 순으로 정렬.(getEnd())
    - 2) 그 외의 경우 따로 리스트에 저장함. 그 중 시작 날짜가 같은 경우
      * 끝 날짜도 같은 경우
        + 시작 시간이 빠른 순으로 정렬
        + 종일 일정이 먼저 나오고 그 후 시간 일정 순.
        + 시간 일정은 시작 시간이 빠른 순으로 정렬 됨.
      * 그 외의 경우(여러 날짜에 걸쳐서 나오는 일정 중 끝 날짜가 같지 않은 경우)
        + 끝 날짜가 늦은 순으로 정렬.
        + 이때는 시간 일정이 먼저 나오고 종일 일정이 나옴.
    - 1), 2) 순으로 최종 리스트에 저장.
  + 종일 일정의 value값은 그 날짜에 오전 9시에 해당하는 값으로 되어 있음.
    - Timezone 때문.
    - 그래서 시작 날짜는 항상 처음으로 오게 하기 위해 그 날짜의 0시에서 1밀리초 만큼 뺀 값으로 설정.
      * 시작, 끝 날짜를 다룰 때 1밀리초 까지는 실제로 저장하지 않으므로 겹칠 일은 없음.
    - 끝 날짜는 하루가 더 더해져 있으므로 하루만큼 빼고 23시 59분 1밀리초의 값을 가지게 함.
      * 이유는???
* 더보기 기능 구현.
  + 월 뷰에서 일정의 개수가 칸의 개수를 넘어가면 넘어간 만큼 더보기에 추가.