💩 Smelly Code Refactoring

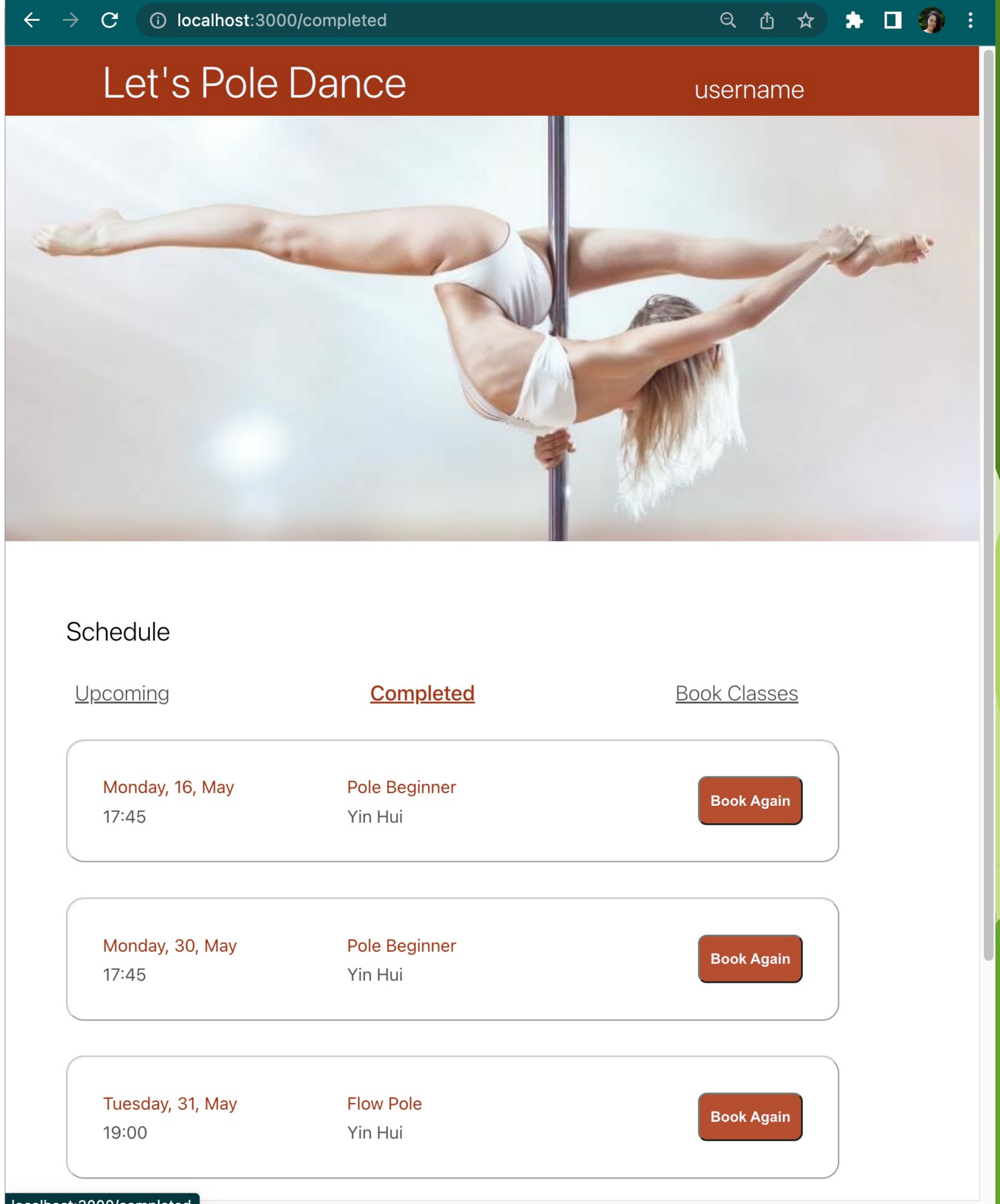
A Real Life Example Trying To Make The Code Less Smelly

→ 🌸

What is a Code Smell?

I'm not completely bad, but I'm not completely good either!

Dance Class Booking App


localhost:3000/completed

Let's Pole Dance

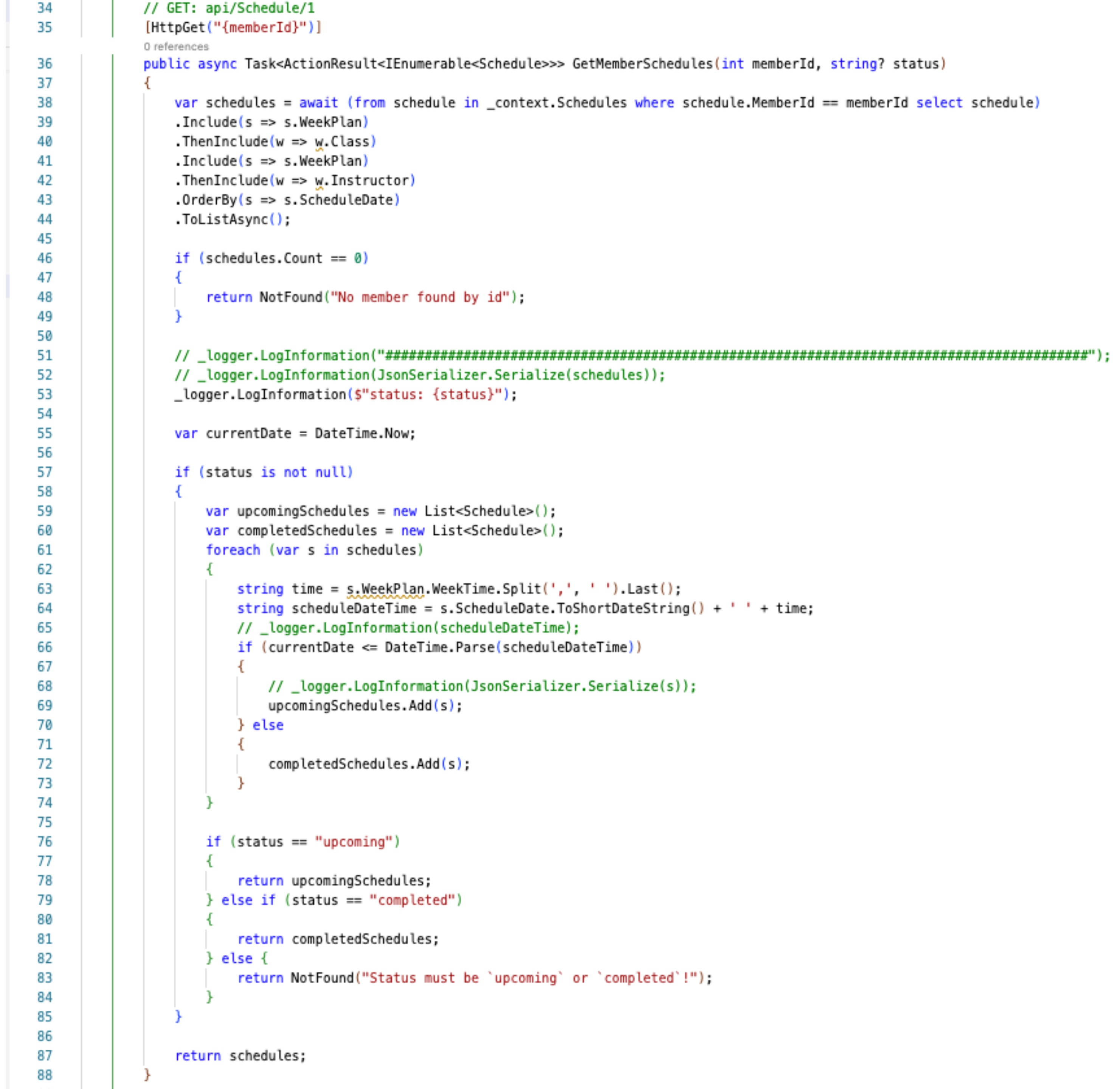
username

Schedule

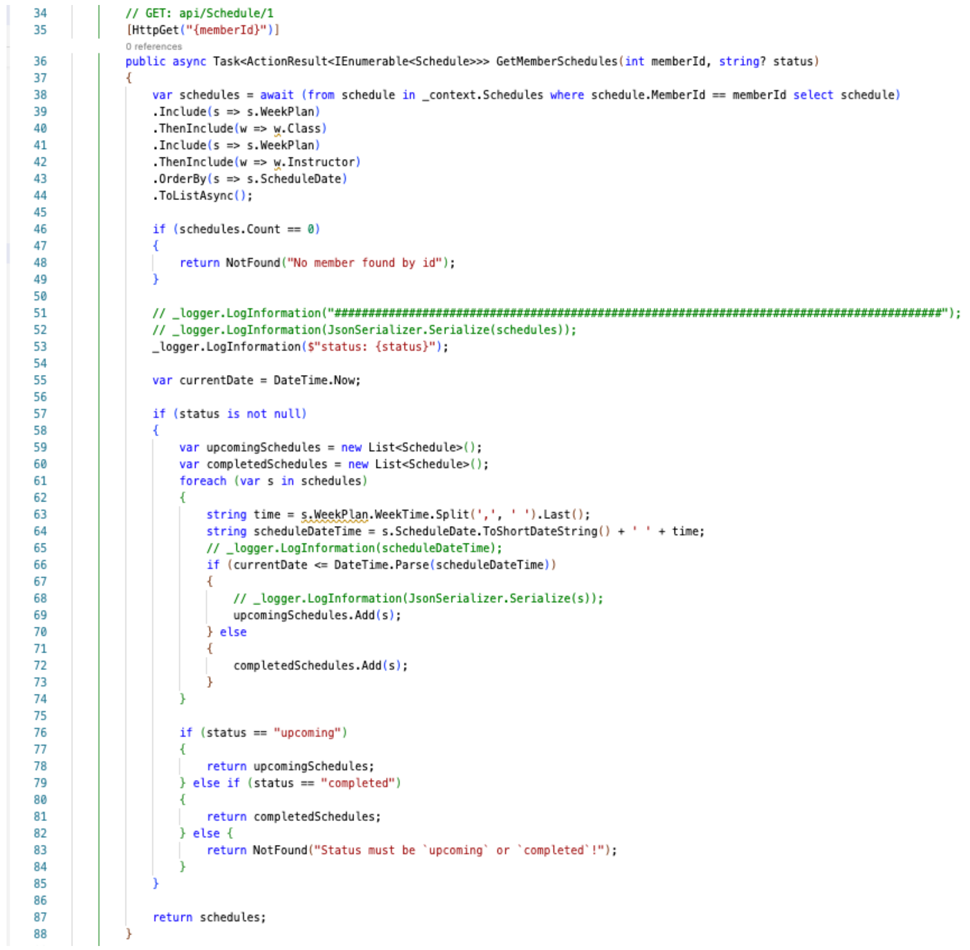
Upcoming | Completed | Book Classes

Monday, 16, May 17:45	Pole Beginner Yin Hui	Book Again
Monday, 30, May 17:45	Pole Beginner Yin Hui	Book Again
Tuesday, 31, May 19:00	Flow Pole Yin Hui	Book Again

Smelly Code

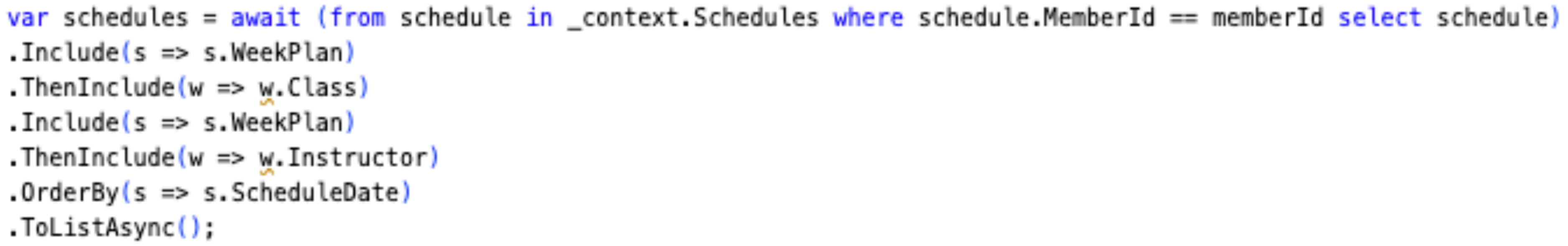


```csharp
// GET: api/Schedule/1
[HttpGet("{memberId}")]
0 references
public async Task<ActionResult<IEnumerable<Schedule>>> GetMemberSchedules(int memberId, string? status)
{
    var schedules = await (from schedule in _context.Schedules where schedule.MemberId == memberId select schedule)
    .Include(s => s.WeekPlan)
    .ThenInclude(w => w.Class)
    .Include(s => s.WeekPlan)
    .ThenInclude(w => w.Instructor)
    .OrderBy(s => s.ScheduleDate)
    .ToListAsync();

    if (schedules.Count == 0)
    {
        return NotFound("No member found by id");
    }

    // _logger.LogInformation("##########################################################################################");
    // _logger.LogInformation(JsonSerializer.Serialize(schedules));
    _logger.LogInformation($"status: {status}");

    var currentDate = DateTime.Now;

    if (status is not null)
    {
        var upcomingSchedules = new List<Schedule>();
        var completedSchedules = new List<Schedule>();
        foreach (var s in schedules)
        {
            string time = s.WeekPlan.WeekTime.Split(',', ' ').Last();
            string scheduleDateTime = s.ScheduleDate.ToShortDateString() + ' ' + time;
            // _logger.LogInformation(scheduleDateTime);
            if (currentDate <= DateTime.Parse(scheduleDateTime))
            {
                // _logger.LogInformation(JsonSerializer.Serialize(s));
                upcomingSchedules.Add(s);
            } else
            {
                completedSchedules.Add(s);
            }
        }

        if (status == "upcoming")
        {
            return upcomingSchedules;
        } else if (status == "completed")
        {
            return completedSchedules;
        } else {
            return NotFound("Status must be `upcoming` or `completed`!");
        }
    }

    return schedules;
}
```



```
// GET: api/Schedule/1
[HttpGet("{memberId}")]
0 references
public async Task<ActionResult<IEnumerable<Schedule>>> GetMemberSchedules(int memberId, string? status)
{
    var schedules = await (from schedule in _context.Schedules where schedule.MemberId == memberId select schedule)
    .Include(s => s.WeekPlan)
    .ThenInclude(w => w.Class)
    .Include(s => s.WeekPlan)
    .ThenInclude(w => w.Instructor)
    .OrderBy(s => s.ScheduleDate)
    .ToListAsync();

    if (schedules.Count == 0)
    {
        return NotFound("No member found by id");
    }

    // _logger.LogInformation("##########################################################################################");
    // _logger.LogInformation(JsonSerializer.Serialize(schedules));
    _logger.LogInformation($"status: {status}");

    var currentDate = DateTime.Now;

    if (status is not null)
    {
        var upcomingSchedules = new List<Schedule>();
        var completedSchedules = new List<Schedule>();
        foreach (var s in schedules)
        {
            string time = s.WeekPlan.WeekTime.Split(',', ' ').Last();
            string scheduleDateTime = s.ScheduleDate.ToShortDateString() + ' ' + time;
            // _logger.LogInformation(scheduleDateTime);
            if (currentDate <= DateTime.Parse(scheduleDateTime))
            {
                // _logger.LogInformation(JsonSerializer.Serialize(s));
                upcomingSchedules.Add(s);
            } else
            {
                completedSchedules.Add(s);
            }
        }

        if (status == "upcoming")
        {
            return upcomingSchedules;
        } else if (status == "completed")
        {
            return completedSchedules;
        } else {
            return NotFound("Status must be `upcoming` or `completed`!");
        }
    }

    return schedules;
}
```

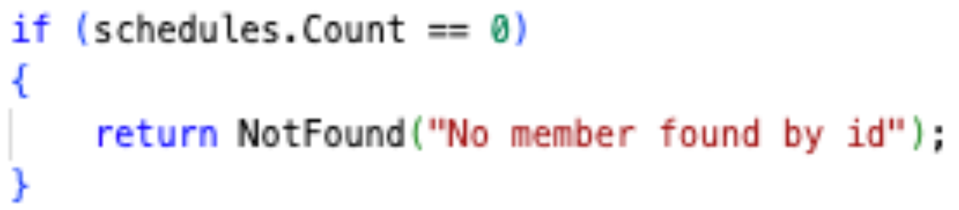
- 💩 Long method
- 💩 Comments



```
var schedules = await (from schedule in _context.Schedules where schedule.MemberId == memberId select schedule)
.Include(s => s.WeekPlan)
.ThenInclude(w => w.Class)
.Include(s => s.WeekPlan)
.ThenInclude(w => w.Instructor)
.OrderBy(s => s.ScheduleDate)
.ToListAsync();
```

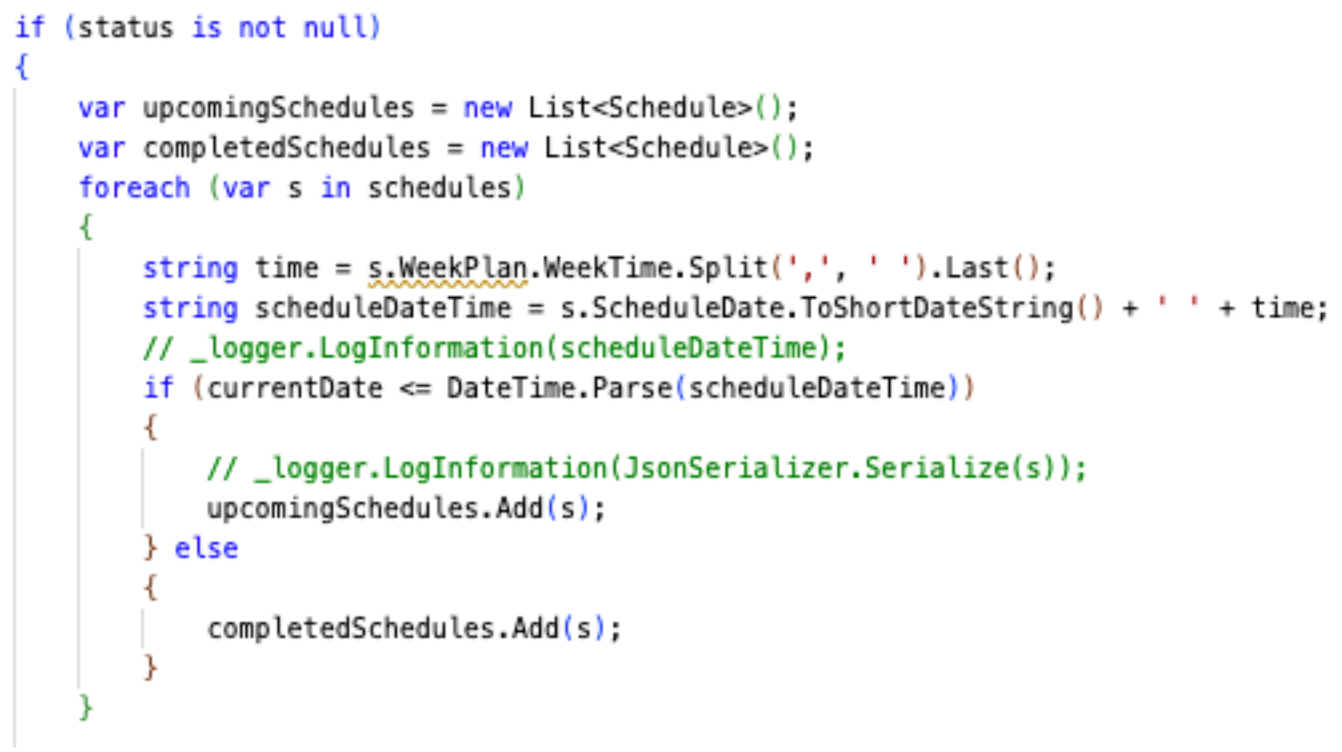
💩 Message Chains

💩 Middle Man



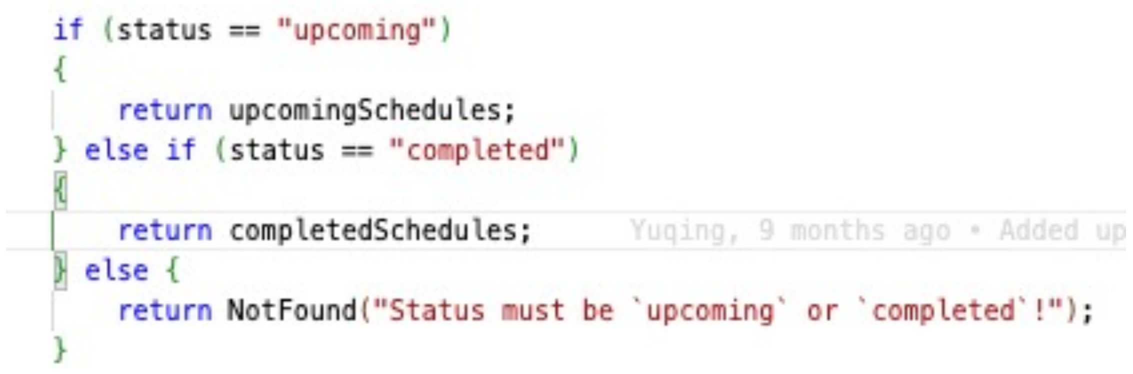
```
if (schedules.Count == 0)
{
    return NotFound("No member found by id");
}
```

💩 Comments / Wrong Error Message Violates “Least Astonishment” (WTF)



```
if (status is not null)
{
    var upcomingSchedules = new List<Schedule>();
    var completedSchedules = new List<Schedule>();
    foreach (var s in schedules)
    {
        string time = s.WeekPlan.WeekTime.Split(',', ' ').Last();
        string scheduleDateTime = s.ScheduleDate.ToShortDateString() + ' ' + time;
        // _logger.LogInformation(scheduleDateTime);
        if (currentDate <= DateTime.Parse(scheduleDateTime))
        {
            // _logger.LogInformation(JsonSerializer.Serialize(s));
            upcomingSchedules.Add(s);
        } else
        {
            completedSchedules.Add(s);
        }
    }
```

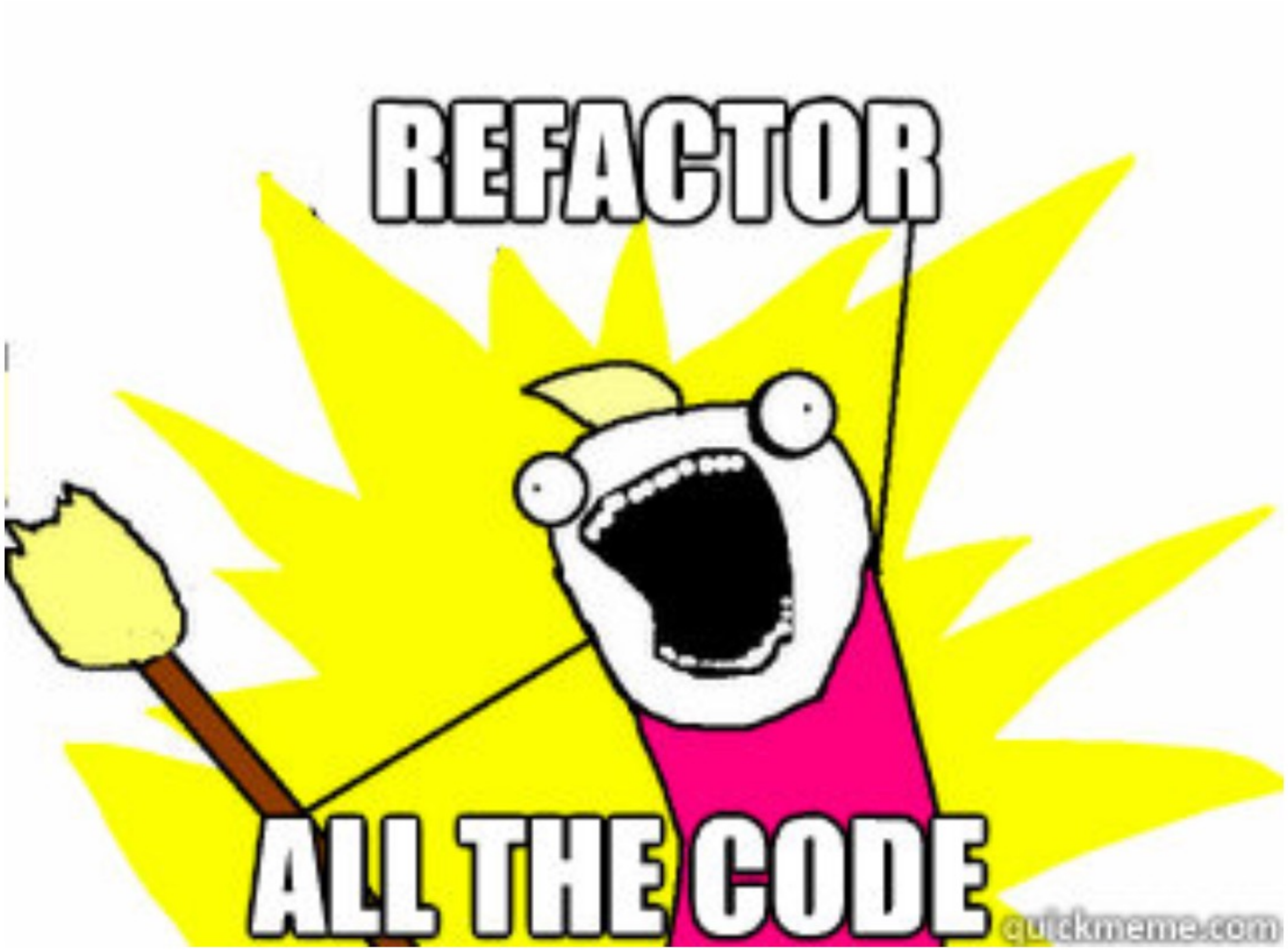
- 💩 Message Chains
- 💩 Data Class
- 💩 Nested Conditional

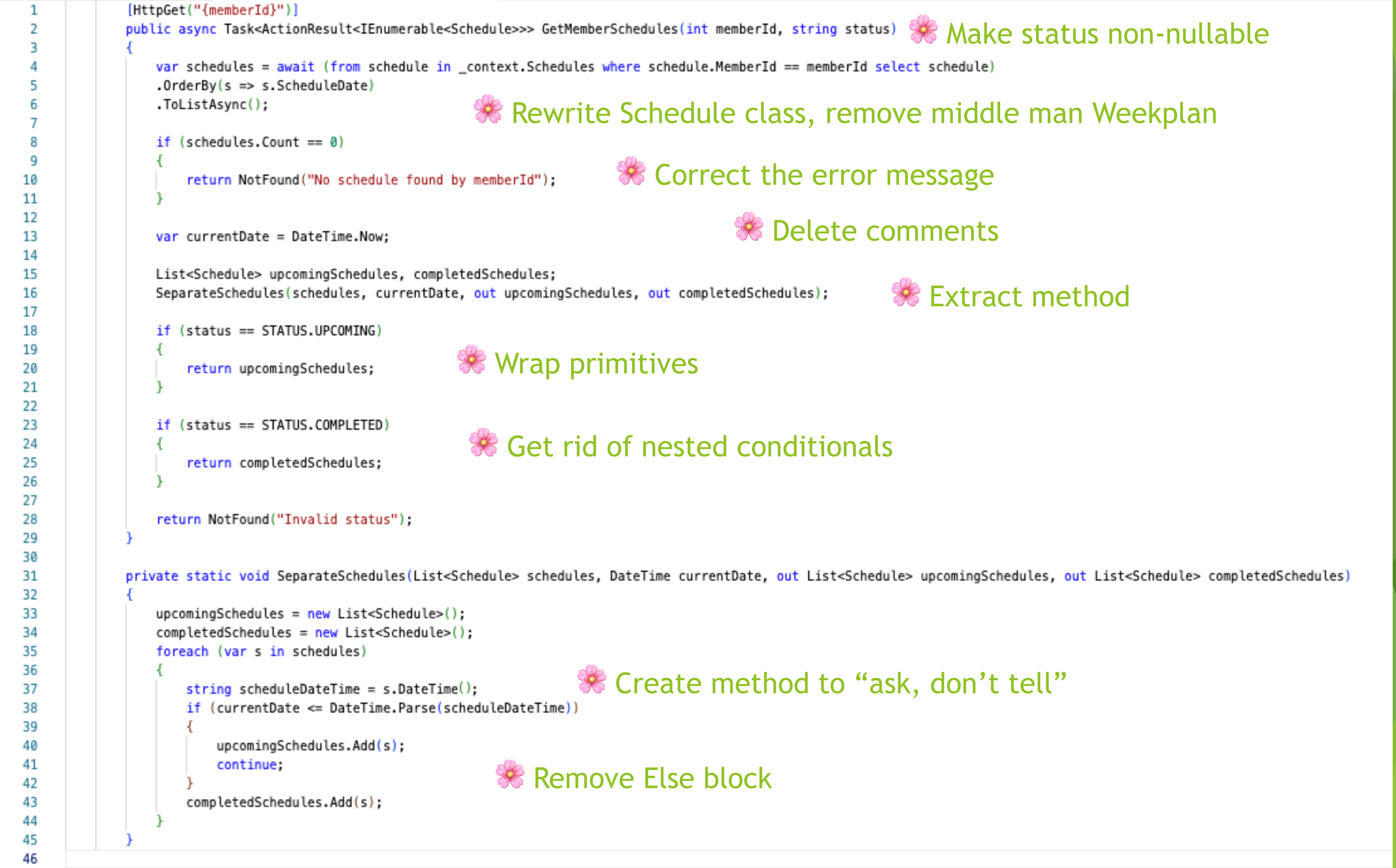


```
if (status == "upcoming")
{
    return upcomingSchedules;
} else if (status == "completed")
{
    return completedSchedules;        Yuqing, 9 months ago • Added up
} else {
    return NotFound("Status must be `upcoming` or `completed`!");
}
```

💩 Primitive Obsession

💩 Else block (object calisthenics)


REFACTOR
ALL THE CODE
quickmeme.com



```
[HttpGet("{memberId}")]
public async Task<ActionResult<IEnumerable<Schedule>>> GetMemberSchedules(int memberId, string status)
{
    var schedules = await (from schedule in _context.Schedules where schedule.MemberId == memberId select schedule)
    .OrderBy(s => s.ScheduleDate)
    .ToListAsync();

    if (schedules.Count == 0)
    {
        return NotFound("No schedule found by memberId");
    }

    var currentDate = DateTime.Now;

    List<Schedule> upcomingSchedules, completedSchedules;
    SeparateSchedules(schedules, currentDate, out upcomingSchedules, out completedSchedules);

    if (status == STATUS.UPCOMING)
    {
        return upcomingSchedules;
    }

    if (status == STATUS.COMPLETED)
    {
        return completedSchedules;
    }

    return NotFound("Invalid status");
}

private static void SeparateSchedules(List<Schedule> schedules, DateTime currentDate, out List<Schedule> upcomingSchedules, out List<Schedule> completedSchedules)
{
    upcomingSchedules = new List<Schedule>();
    completedSchedules = new List<Schedule>();
    foreach (var s in schedules)
    {
        string scheduleDateTime = s.DateTime();
        if (currentDate <= DateTime.Parse(scheduleDateTime))
        {
            upcomingSchedules.Add(s);
            continue;
        }
        completedSchedules.Add(s);
    }
}
```

🌸 Make status non-nullable

🌸 Rewrite Schedule class, remove middle man Weekplan

🌸 Correct the error message

🌸 Delete comments

🌸 Extract method

🌸 Wrap primitives

🌸 Get rid of nested conditionals

🌸 Create method to “ask, don’t tell”

🌸 Remove Else block

Thank you very much!

Grazie Mille!

Author: Yuqing Lu

Contact: luyuqing0708@gmail.com