Лабораторная работа 2 Жаркова Е.С.

Условно лабораторную работу можно разделить на 3 части:

- 1) Написание сервера на Java
- 2) Написание парсера данных
- 3) Написание генератора

## Часть 1. Сервер

- Смотрим про SpringBoot: https://www.youtube.com/playlist?list=PLwvrYc4311MzeA2bBYQhCWr2gvWLs9A7S
- 2) Разбираемся с javax.persistence.\*: <a href="https://javastudy.ru/spring-data-jpa/annotation-persistence/">https://javastudy.ru/spring-data-jpa/annotation-persistence/</a>
  - 3) И еще, на самом деле, много с чем разбираемся по ходу дела.
  - 4) Радуемся, когда GET и POST запросы начинают работать и приступаем к следующей части.

### Часть 2. Парсер

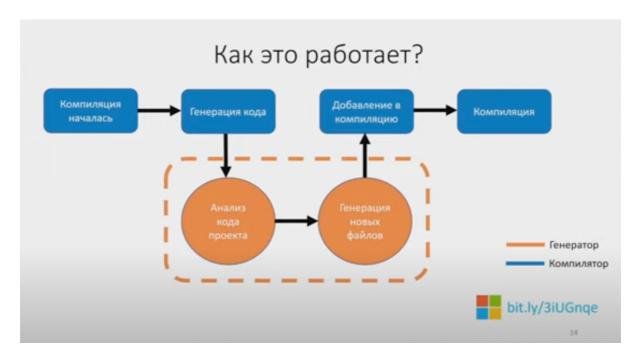
Самая вкусная часть лабораторной. Просто работаешь со строками и получаешь удовольствие.

Суть: пробегаем по строке и по порядку берем то, что нам нужно, потом удаляем это, не забываем при этом чистить пробелы при помощи .Trim(), а при парсинге типа аргумента нужно не забыть .ToLower

### Часть 3. Генератор

# Так что такое эти генераторы?

- Новая технология метапрограммирования от Microsoft
- Часть процесса компиляции
- Доступ к модели вашего кода
- Результат генератора новые файлы



# Этапы работы:

- 1) Отрицаем генератор напишется как-то сам, не надо его трогать
- 2) Гнев дедлайн уже близко, а генератор еще не написал себя сам
- 3) Торг уговариваем самого себя сесть и написать уже
- 4) Депрессия плачем, потому что понимаем, что не успеем написать
  - < Знак свыше перенос дедлайна >
- 5) Принятие появляются новые силы, начинаем писать генератор
- 6) Читаем про Roslyn, Generator
- 7) Идем на <a href="https://roslynquoter.azurewebsites.net">https://roslynquoter.azurewebsites.net</a> и генерируем много непонятного (поначалу) кода
- 8) Начинаем в нем разбираться
- 9) И вместо статических значений вставляем нужные нам переменные, которые достали парсером
- 10) Все рушится, мы пытаемся ловить ошибки и исправлять
- 11) Все готово, радуемся

Боль настигла на моментах: подключения Postgres'а и написания генератора. Но если для решения первой проблемы достаточно было найти человека с маком, который уже прошел по этой тропе ада и может объяснить все не на английском языке с индусским акцентом, то для решения второй пришлось выискивать вручную ошибку (дебага-то нет) Проблема возникла в генерации аргументов методов, если они существуют – все рушится

#### Код был таков:

<Работа с httpRequest с параметрами>

### Код сейчас таков:

```
int countParameter = 0;
SyntaxNodeOrToken[] argArray = new SyntaxNodeOrToken[methodDeclaration.ArgList.Count];
string newUrl = methodDeclaration.Url + "?";
foreach (var arg in methodDeclaration.ArgList)
{
    var parameter = SyntaxFactory.Parameter(
        SyntaxFactory.Identifier(arg.Name)
    )
    .WithType(
        SyntaxFactory.IdentifierName(arg.Type)
    ).NormalizeWhitespace();
    argArray[countParameter] = parameter;
    newUrl = newUrl + $"{arg.Name} = {{{arg.Name}}}&";
    countParameter++;
}
```

Вывод: хочешь усложнить? Подумай и передумай.