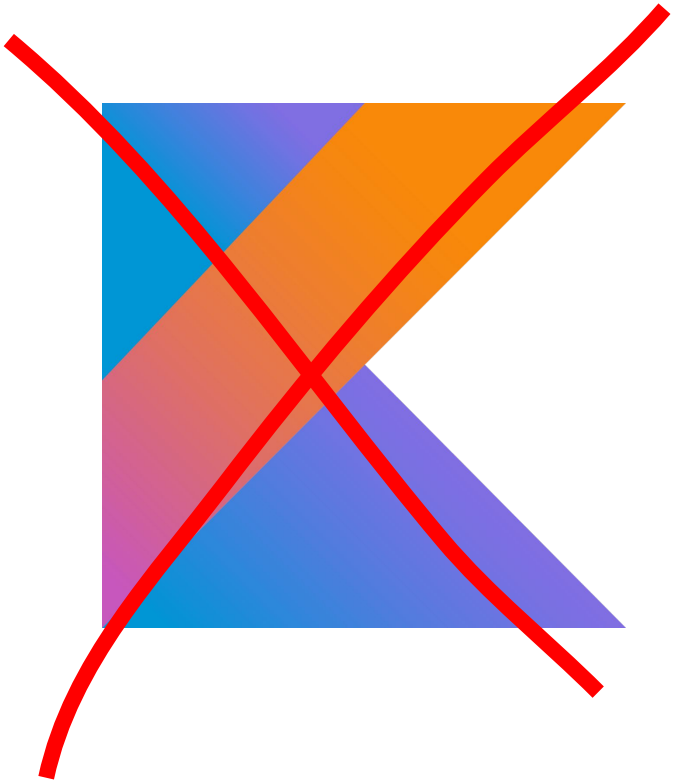
Fluent API на Java

Артём Бояршинов

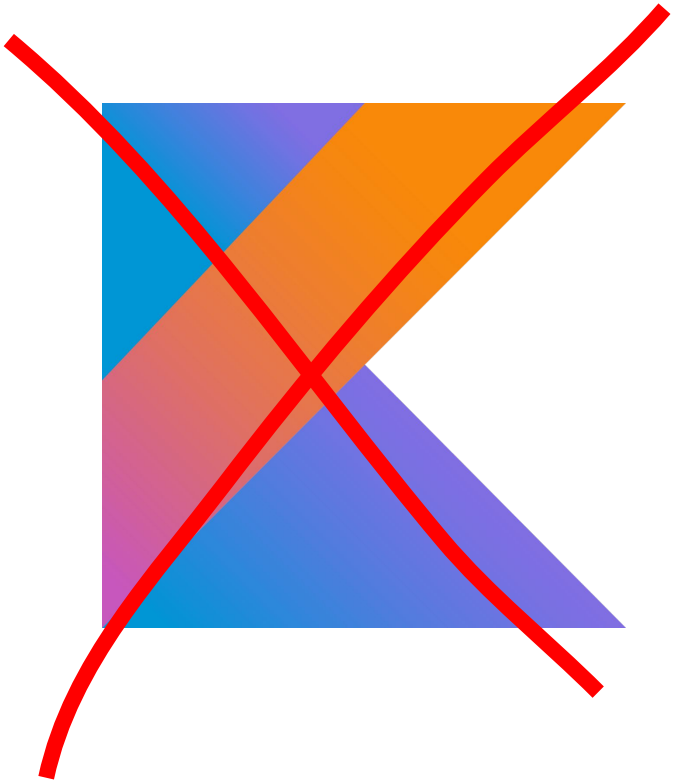
- Software developer
- СБП - Система быстрых платежей
- Член ПК Podlodka Java Crew

https://github.com/Boiarshinov/fluent-api-demo

Disclaimer

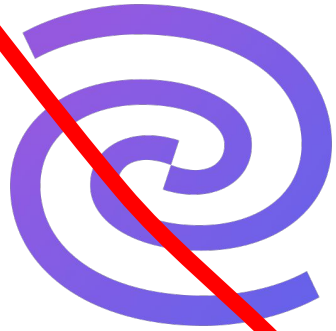


Disclaimer



GitHub Copilot

JetBrains AI Assistant



Imperative Java VS Fluent API

```java
Instant start = Instant.now();
Duration timeout = Duration.ofSeconds(10);
do {
    Thread.sleep(200);
    var entity = repo.get("id");
    if ("EXPECTED".equals(entity.status)) {
        return;
    }
} while (Instant.now().isBefore(start.plus(timeout)));
throw new AssertionError("Status was not updated to EXPECTED");
```

```java
Awaitility.await("Entity status should be updated to EXPECTED")
    .atMost(Duration.ofSeconds(10))
    .pollDelay(Duration.ofMillis(200))
    .until(() -> "EXPECTED".equals(repo.get("id").status));
```

Fluent API - это API, реализующий внутренний DSL, нацеленный на удобство чтения и использования.

Martin Fowler

https://martinfowler.com/bliki/FluentInterface.html

Fluent API

- Где используется?
- Как написать?
- Когда писать?

Где используется

- Поэтапное создание объектов
- Конфигурация
- Тесты, ассерты

Где используется

OkHttp3

```
1  HttpUrl url = new HttpUrl.Builder()
2      .scheme("https")
3      .host("github.com")
4      .port(443)
5      .encodedPath("Boiarshinov")
6      .addQueryParameter("key", "value")
7      .addQueryParameter("key2", "value2")
8      .build();
```

- Поэтапное создание объектов
- Конфигурация
- Тесты, ассерты

Где используется

Spring Security

```
1  public SecurityFilterChain config(HttpSecurity http) {
2      return http
3          .httpBasic()
4          .and()
5          .authorizeRequests()
6              .requestMatchers("/").permitAll()
7              .requestMatchers("/api/*").authenticated()
8              .requestMatchers("/maintain/*").hasRole("ADMIN")
9              .requestMatchers("/actuator/health").permitAll()
10         .and()
11         .csrf()
12             .disable()
13         .exceptionHandling()
14             .accessDeniedPage("/not_authorized")
15         .and()
16         .build();
17 }
```

- Поэтапное создание объектов
- Конфигурация
- Тесты, ассерты

Где используется

Kafka Streams

- Поэтапное создание объектов
- Конфигурация
- Тесты, ассерты

```
1  public Topology createTopology() {
2      return new Topology()
3          .addSource("input", Topics.inputTopic1)
4
5          .addProcessor("parser", parserProcessor, "input")
6          .addProcessor("createTimeFilter", createTimeFilterProcessor, "parser")
7          .addProcessor("flyDataProcessor", flyDataProcessor, "createTimeFilter")
8          .addProcessor("meatballDataProcessor", meatballDataProcessor, "createTimeFilter")
9          .addProcessor("validation", validationProcessor, "flyDataProcessor", "meatballDataProcessor")
10         .addProcessor("mapper", messageMapperProcessor, "validation")
11         .addProcessor("paymentLink", paymentLinkProcessor, "mapper")
12         .addProcessor("transaction", transactionProcessor, "mapper")
13
14         .addSink("paymentLinkStore", Topics.outputTopic1, "paymentLink")
15         .addSink("transactionStore", Topics.outputTopic2, "transaction");
16 }
```

Где используется

Awaitility

- Поэтапное создание объектов
- Конфигурация
- Тесты, ассерты

```
1  Awaitility.await("Entity status should be updated to EXPECTED")
2      .atMost(Duration.ofSeconds(10))
3      .pollDelay(Duration.ofMillis(200))
4      .until(() -> "EXPECTED".equals(repo.get("id").status));
```

Где используется

MockMvc

- Поэтапное создание объектов
- Конфигурация
- Тесты, ассерты

```
1  mockMvc.perform(post("/path/to/resource")
2          .contentType(MediaType.APPLICATION_JSON)
3          .content(objectMapper.writeValueAsString(requestBodyDto))
4          .header("X-CUSTOM", "abcd"))
5      .andDo(print())
6      .andExpect(status().isCreated())
7      .andExpect(jsonPath("$.code", is("C000")))
8      .andExpect(jsonPath("$.message", is("OK")))
9      .andExpect(jsonPath("$.data.id", is(1)));
```

Пример

T-Bank Прокси / Gateway

T-Bank Прокси / Gateway. Фичи

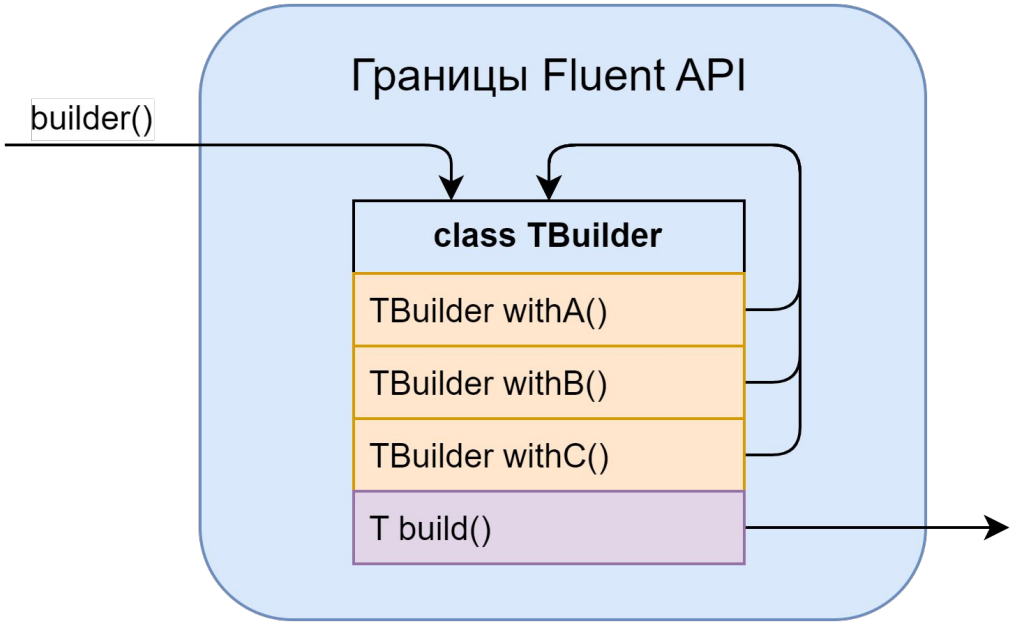
- Список получателей
- Балансировка запросов по получателям
- Контроль request’ов
  - Проверка входящего контента на вхождение в черный список
- Контроль respons’ов
  - Замена части исходящего контента

Как написать Fluent API

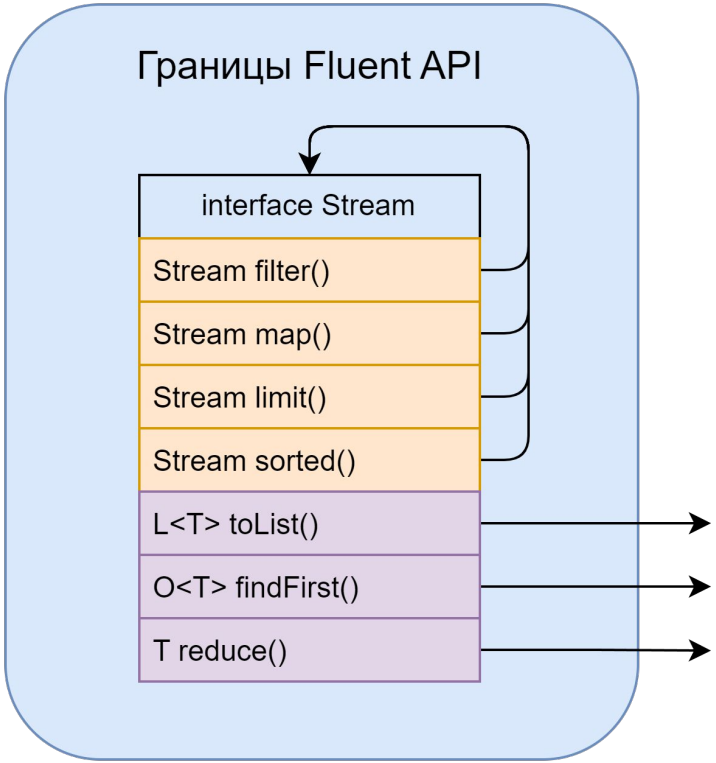
Приёмы Fluent API

- Method chaining - последовательный вызов методов
- Step interfaces - шаги, регулирующие последовательность и обязательность вызова методов
- Смена контекста
  - С помощью method chaining
  - С помощью лямбда-выражений
- Selftypes - самообобщение

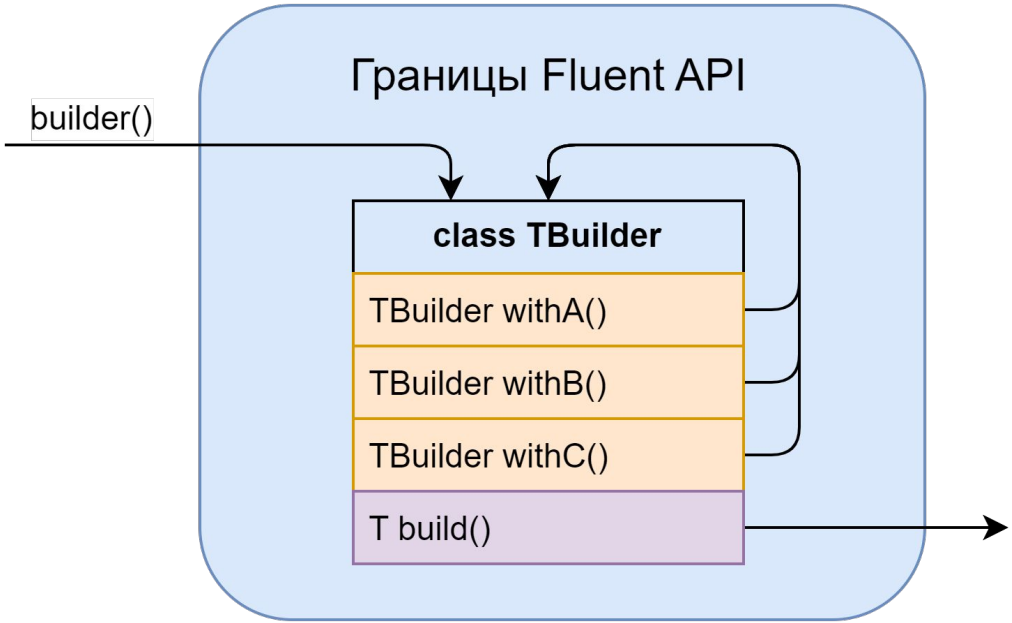
Builder


Границы Fluent API
builder()
class TBuilder
TBuilder withA()
TBuilder withB()
TBuilder withC()
T build()

Stream


Границы Fluent API
interface Stream
Stream filter()
Stream map()
Stream limit()
Stream sorted()
L<T> toList()
O<T> findFirst()
T reduce()

Builder

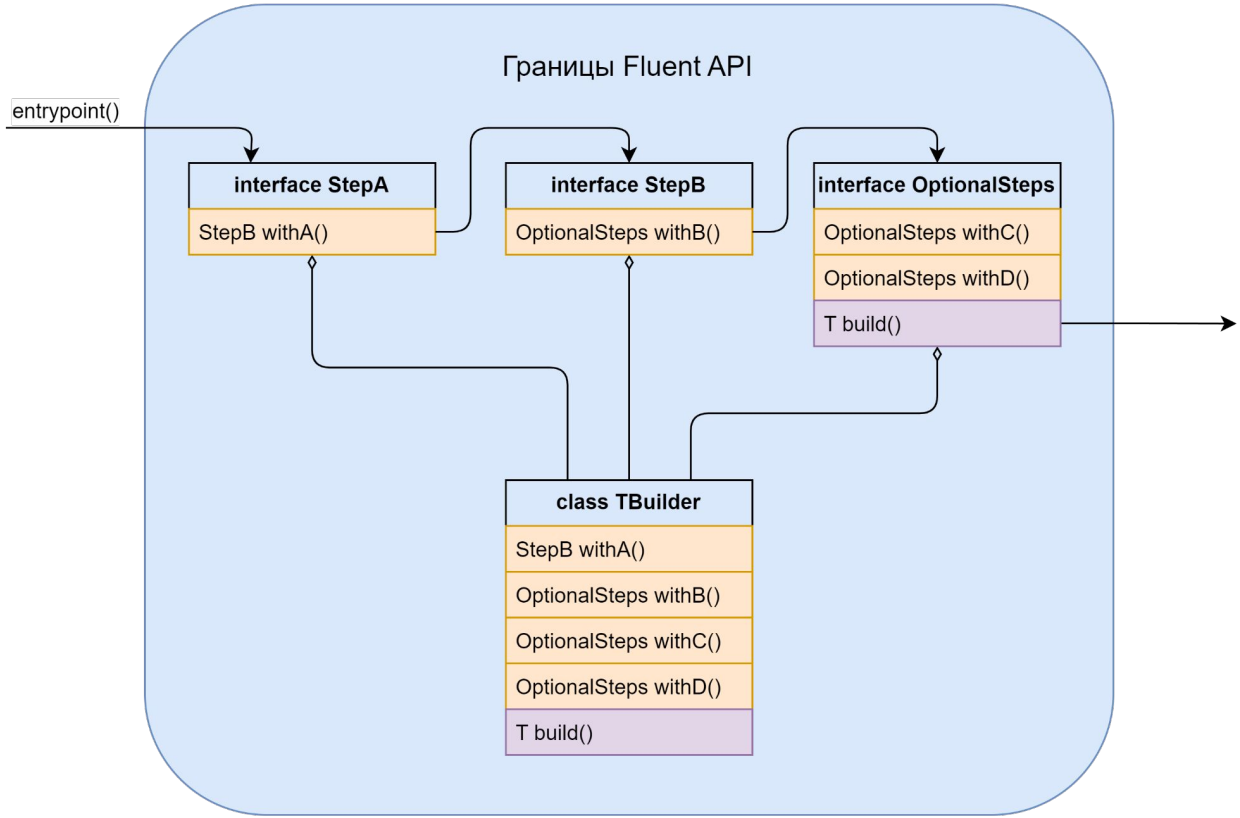


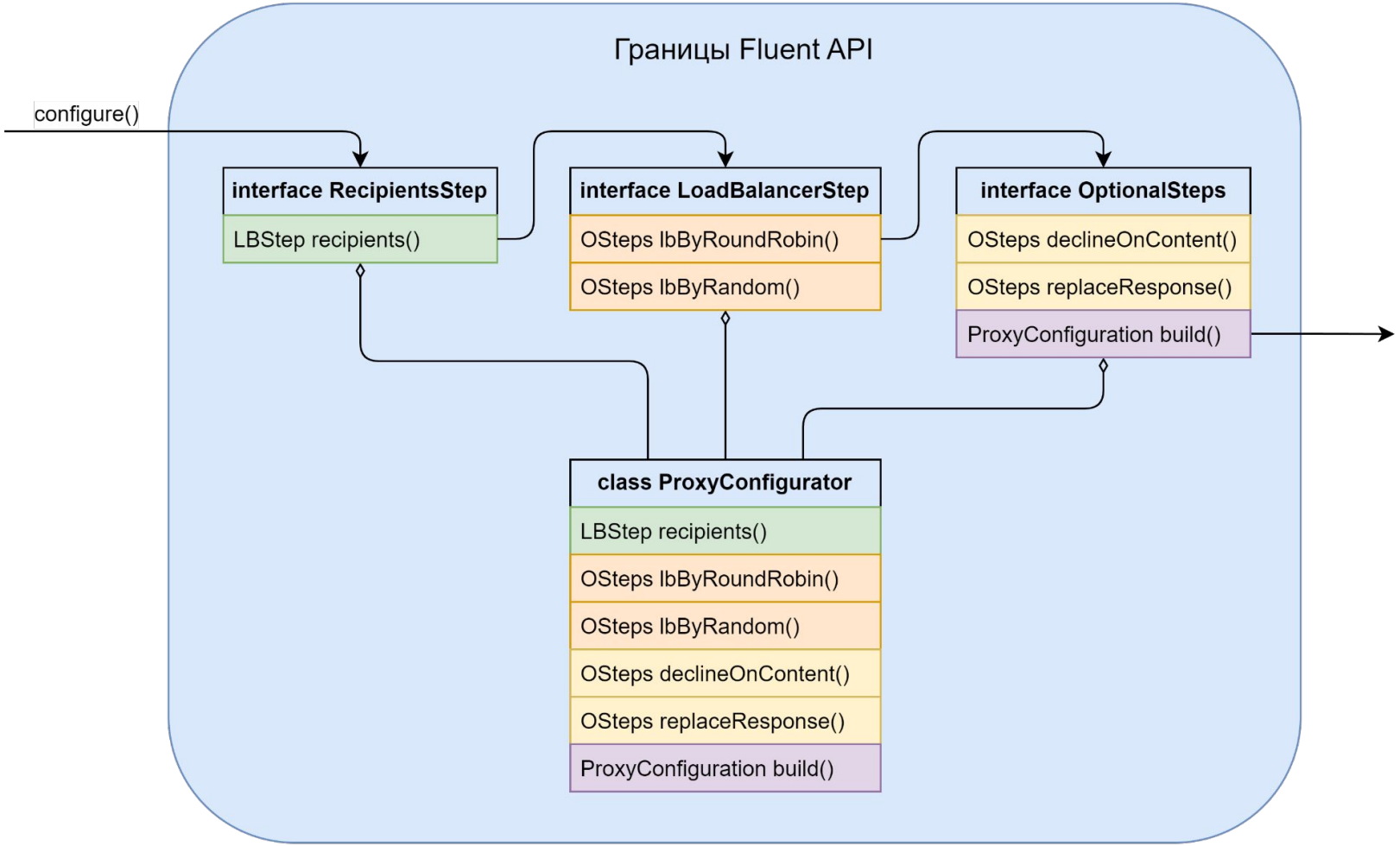
- можно забыть заполнить обязательные поля
- случайно можно заполнить одно поле дважды

Приёмы Fluent API

- Method chaining - последовательный вызов методов
- Step interfaces - шаги, регулирующие последовательность и обязательность вызова методов
- Смена контекста
  - С помощью method chaining
  - С помощью лямбда-выражений
- Selftypes - самообобщение

Step Interfaces



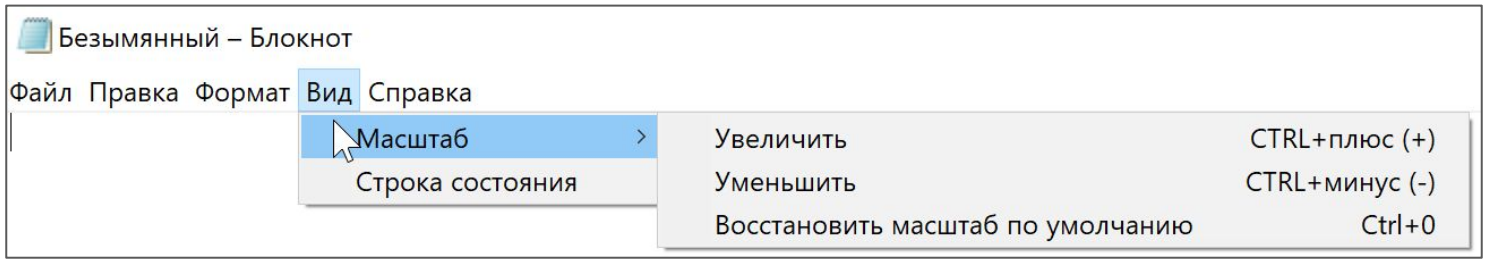

Границы Fluent API
configure()
interface RecipientsStep
LBStep recipients()
interface LoadBalancerStep
OSteps lbByRoundRobin()
OSteps lbByRandom()
interface OptionalSteps
OSteps declineOnContent()
OSteps replaceResponse()
ProxyConfiguration build()
class ProxyConfigurator
LBStep recipients()
OSteps lbByRoundRobin()
OSteps lbByRandom()
OSteps declineOnContent()
OSteps replaceResponse()
ProxyConfiguration build()

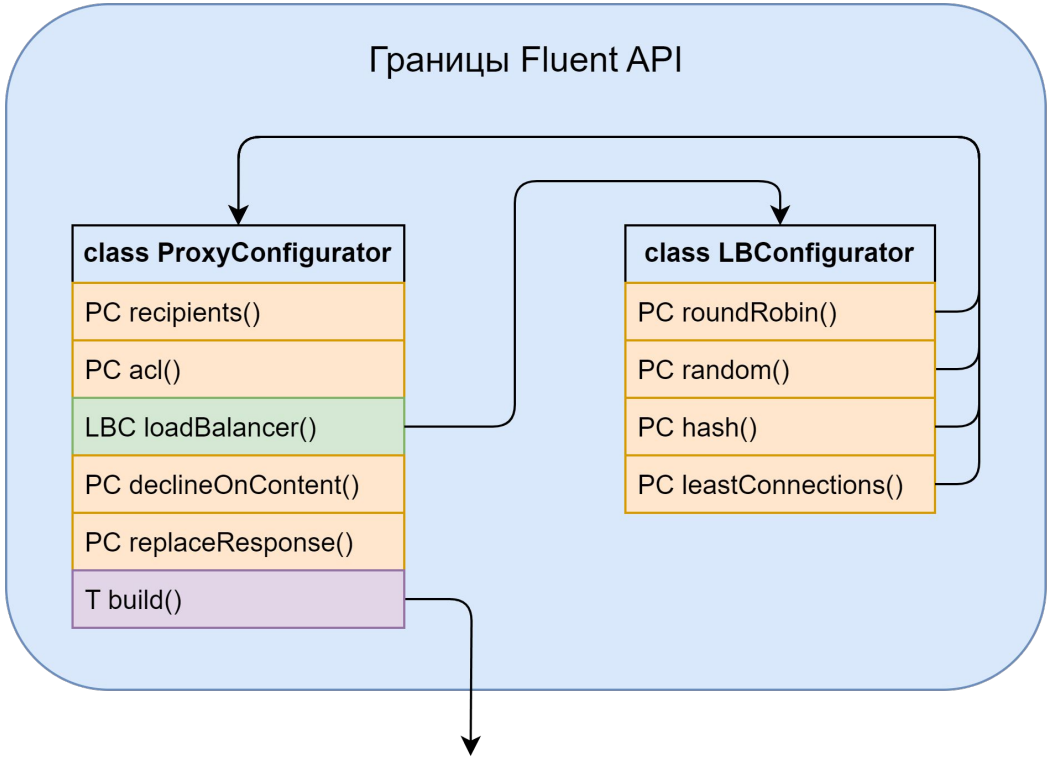
Приёмы Fluent API

- Method chaining - последовательный вызов методов
- Step interfaces - шаги, регулирующие последовательность и обязательность вызова методов
- Смена контекста
  - С помощью method chaining
  - С помощью лямбда-выражений
- Selftypes - самообобщение

Fluent API позволяет не знать API, но при этом пользоваться им

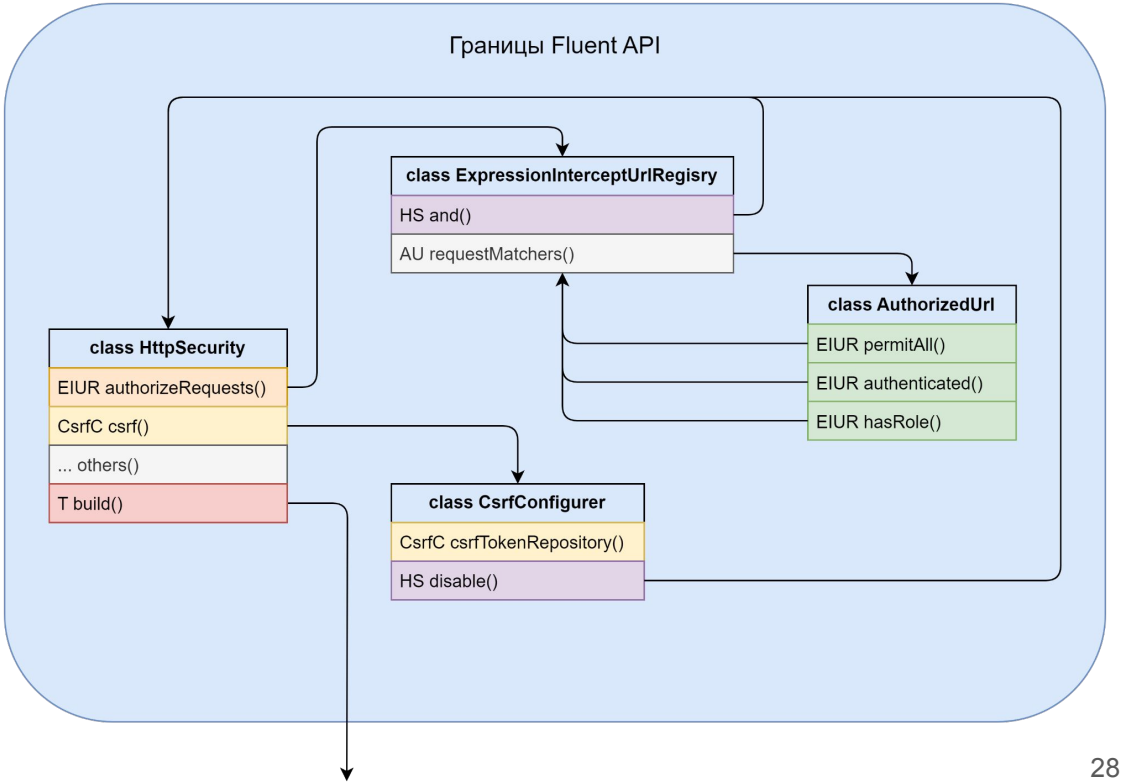
(с помощью подсказок IDE)



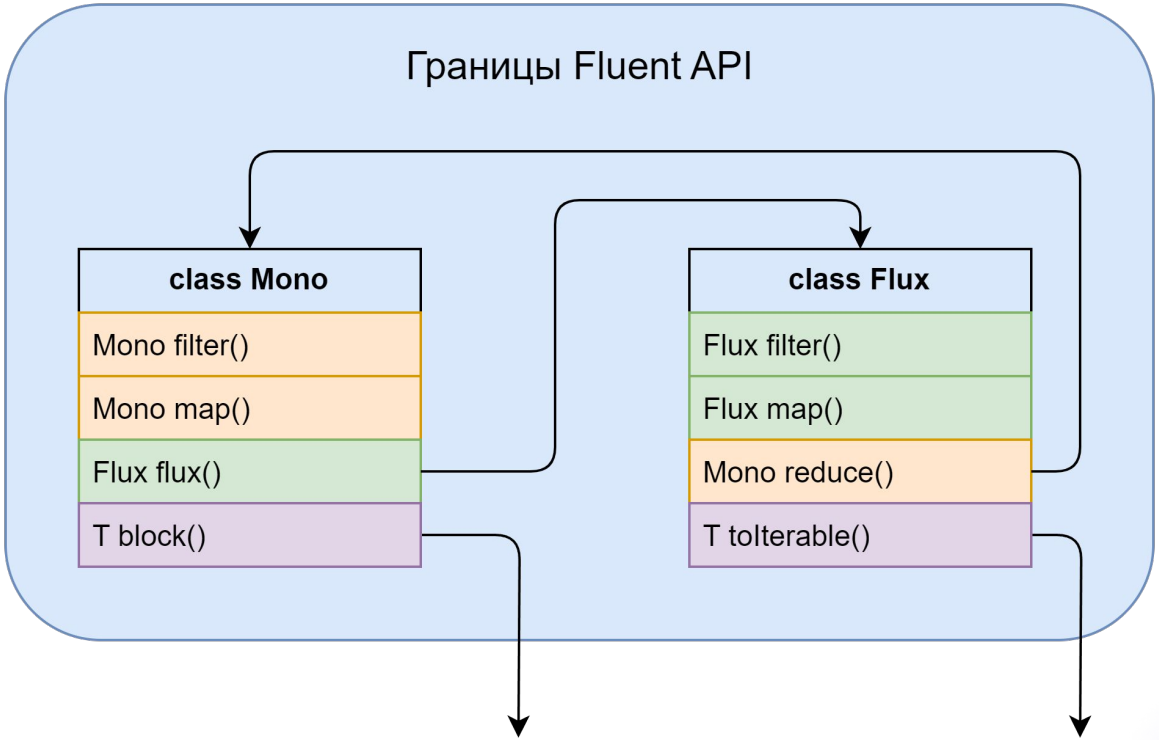

Границы Fluent API
class ProxyConfigurator
PC recipients()
PC acl()
LBC loadBalancer()
PC declineOnContent()
PC replaceResponse()
T build()
class LBConfigurator
PC roundRobin()
PC random()
PC hash()
PC leastConnections()

org.springframework.security:spring-security-config

```
1  public SecurityFilterChain config(HttpSecurity http) {
2      return http
3          .httpBasic()
4          .and()
5          .authorizeRequests()
6              .requestMatchers("/").permitAll()
7              .requestMatchers("/api/*").authenticated()
8              .requestMatchers("/maintain/*").hasRole("ADMIN")
9              .requestMatchers("/actuator/health").permitAll()
10         .and()
11         .csrf()
12             .disable()
13         .exceptionHandling()
14             .accessDeniedPage("/not_authorized")
15         .and()
16         .build();
17 }
```



io.projectreactor



Смена контекста

```
1  public SecurityFilterChain config(HttpSecurity http) {
2      return http
3          .httpBasic()
4          .and()
5          .authorizeRequests()
6              .requestMatchers("/").permitAll()
7              .requestMatchers("/api/*").authenticated()
8              .requestMatchers("/maintain/*").hasRole("ADMIN")
9              .requestMatchers("/actuator/health").permitAll()
10         .and()
11         .csrf()
12             .disable()
13         .exceptionHandling()
14             .accessDeniedPage("/not_authorized")
15         .and()
16         .build();
17 }
```

- Для визуализации смены контекста нужны отступы
- Отступы пропадают при автоформатировании
- Паразитные методы and() для переходов между контекстами

Приёмы Fluent API

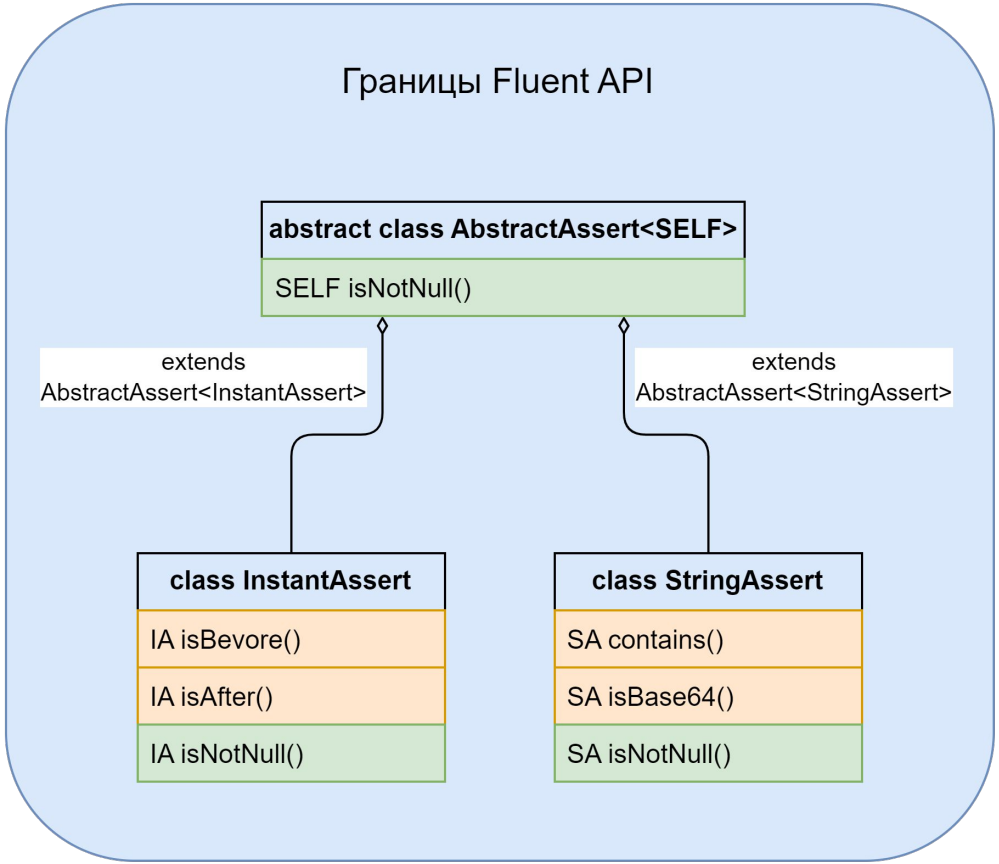
- Method chaining - последовательный вызов методов
- Step interfaces - шаги, регулирующие последовательность и обязательность вызова методов
- Смена контекста
  - С помощью method chaining
  - С помощью лямбда-выражений
- Selftypes - самообобщение

```java
public SecurityFilterChain config(HttpSecurity http) {
    return http
        .httpBasic()
        .and()
        .authorizeRequests()
            .requestMatchers("/").permitAll()
            .requestMatchers("/api/*").authenticated()
            .requestMatchers("/maintain/*").hasRole("ADMIN")
            .requestMatchers("/actuator/health").permitAll()
        .and()
        .csrf()
            .disable()
        .exceptionHandling()
            .accessDeniedPage("/not_authorized")
        .and()
        .build();
}
```

```java
// Начиная с Spring Security 5.5
public SecurityFilterChain config(HttpSecurity http) {
    return http
        .httpBasic(Customizer.withDefaults())
        .authorizeHttpRequests(requests -> requests
            .requestMatchers("/").permitAll()
            .requestMatchers("/api/*").authenticated()
            .requestMatchers("/maintain/*").hasRole("ADMIN")
            .requestMatchers("/actuator/health").permitAll()
        )
        .csrf(csrf -> csrf.disable())
        .exceptionHandling(exConf -> exConf.accessDeniedPage("/not_authorized"))
        .build();
}
```

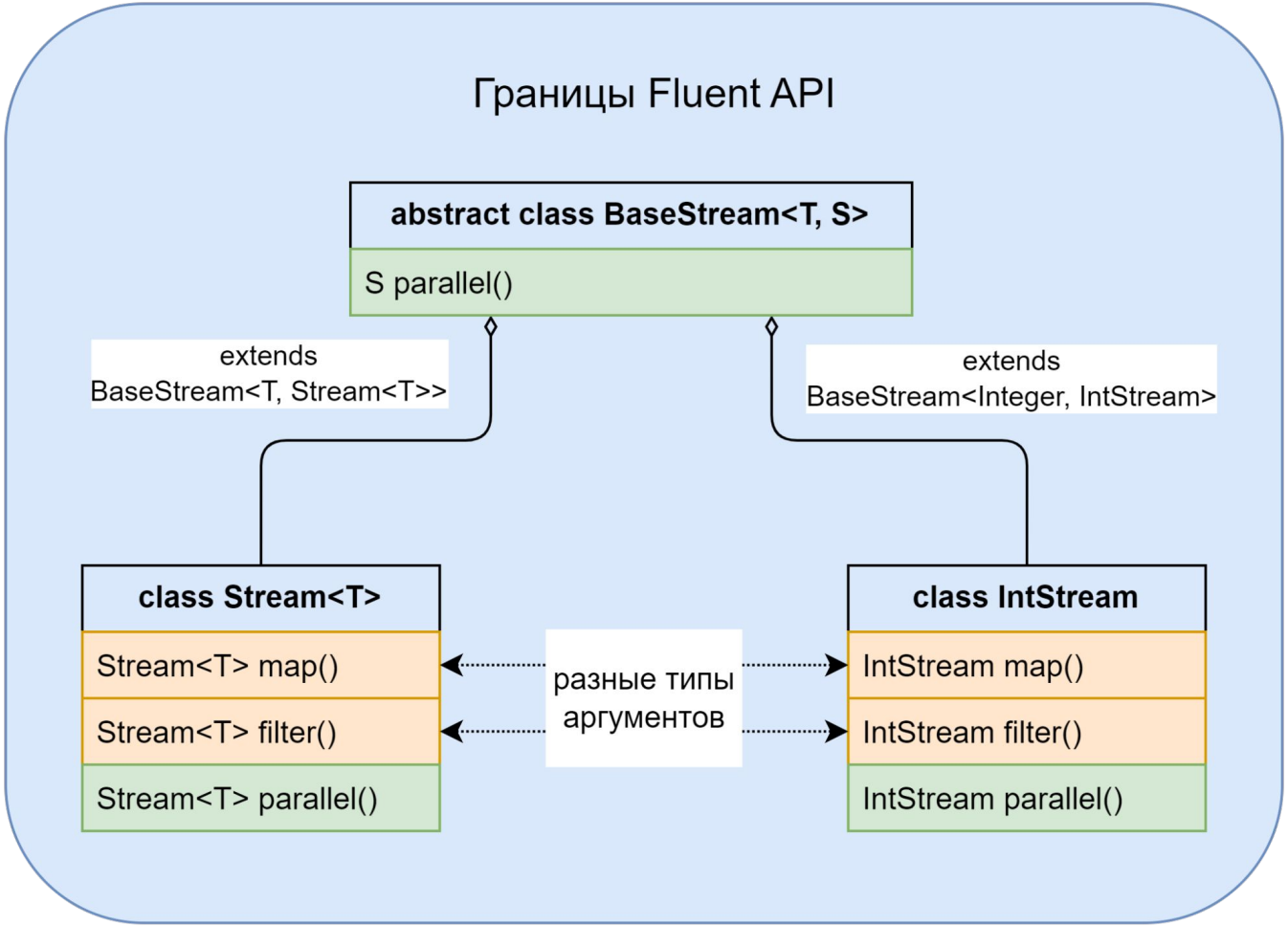
Приёмы Fluent API

- Method chaining - последовательный вызов методов
- Step interfaces - шаги, регулирующие последовательность и обязательность вызова методов
- Смена контекста
  - С помощью method chaining
  - С помощью лямбда-выражений
- Selftypes - самообобщение



AssertJ

https://web.archive.org/web/20090807111719/http://blogs.sun.com/ahe/entry/selftypes



java.util.stream

Когда писать?

Когда писать?

- Фреймворк
- Либа для многих команд
- PR в Lombok для обязательных полей в Builder :)

Приёмы Fluent API

- Method chaining - последовательный вызов методов
- Step interfaces - шаги, регулирующие последовательность и обязательность вызова методов
- Смена контекста
  - С помощью method chaining
  - С помощью лямбда-выражений
- Selftypes - самообобщение
- …

Спасибо за внимание!