Как и когда использовать порождающий шаблон проектирования «**Строитель**»

Содержание

Знакомство с кодом проекта

1. Ознакомтесь с кодом проекта и проблемой **problem/index.ts, app.ts**

Задание 1. Выделение кода в класс

1. Создайте файл **src/step1/index.ts** и добавьте в него следующий фрагмент кода:

```
export class EventFilterBuilder {
    // a private filterResult object to represent the internal state
    private filterResult: { [key: string]: string } = {};

    // “set” method to set the individual filter
    setCategoryFilter(category: string): this {
        if (category) {
            this.filterResult.categoryFilter = `(category = '${category}')`;
        }
        return this;
    }

    // “set” method to set the individual filter
    setOpenOnlyFilter(openOnly: boolean): this {
        if (openOnly) {
            this.filterResult.openOnlyFilter = `(deleted = 0)`;
        }
        return this;
    }

    // “set” method to set the individual filter
    setStatusFilter(status: string | string[]): this {
        const statusList = Array.isArray(status) ? status : [status];

        this.filterResult.statusFilter = `(${this.buildFilterFromArray(
            statusList,
            'status'
        )})`;
        return this;
    }

    // a build method to generate the filter output
    build(): string {
        const filters = Object.values(this.filterResult).filter(Boolean);

        return filters?.length > 0 ? filters.join(` and `) : '';
    }

    private buildFilterFromArray(values: string[], prop: string) {
        return values?.map(e => `${prop} = '${e}'`).join(` or `) ?? '';
    }
}
```

2. Внесите изменения в файл **app.ts**. Используйте следующий фрагмент кода:

```
// 1
import { EventFilterBuilder } from './step1';

// 2
const filterResult = new EventFilterBuilder()
        .setCategoryFilter(orderFilter.category)
        .setStatusFilter(orderFilter.status)
        .setOpenOnlyFilter(true)
        .build();
    console.log(filterResult);
```

3. Запустите проект, посмотрите результат работы приложения в консоли браузера

Задание 2. Реализация неизменяемого внутреннего состояния

1. Внесите изменения в файл **app.ts**. Используйте следующий фрагмент кода:

```
// problem with mutation of filterResult
const openOnlyBuilder = new EventFilterBuilder().setOpenOnlyFilter(true);
const statusBuilder = openOnlyBuilder.setStatusFilter(orderFilter.status);
console.log('openOnlyBuilder result:', openOnlyBuilder.build()); // (status =
'apporved' or status = 'paid') and (deleted = 0)
console.log('statusBuilder result:', statusBuilder.build()); //  (status =
'apporved' or status = 'paid') and (deleted = 0)
```

2. Запустите проект, посмотрите результат работы приложения в консоли браузера.
3. Создайте файл **src/step2/index.ts** и скопируйте в него содержимое файла **src/step1/index.ts**
4. Внесите изменения в класс **EventFilterBuilder** в файле **src/step2/index.ts** используя следующий фрагмент кода:

```
// 1
private filterResult: { readonly [key: string]: string } = {};

// 2
constructor(current = {}) {
  this.filterResult = current;
}

// 3
setCategoryFilter(category: string): this EventFilterBuilder {
        if (category) {
            this.filterResult.categoryFilter = `(category = '${category}')`;
        }
        return this;
      return new EventFilterBuilder({
            ...this.filterResult,
            category: category ? `(category = '${category}')` : undefined,
        });
}

// 4
setOpenOnlyFilter(openOnly: boolean): this EventFilterBuilder{
        if (openOnly) {
            this.filterResult.openOnlyFilter = `(deleted = 0)`;
        }
        return this;
      return new EventFilterBuilder({
            ...this.filterResult,
            openOnlyFilter: openOnly ? `(deleted = 0)` : undefined,
```

```
        });
    }

// 5
setStatusFilter(status: string | string[]): this EventFilterBuilder{
        const statusList = Array.isArray(status) ? status : [status];

        this.filterResult.statusFilter = `(${this.buildFilterFromArray(
            statusList,
            'status'
        )})`;
        return this;

        return new EventFilterBuilder({
            ...this.filterResult,
            statusFilter: `(${this.buildFilterFromArray(
                statusList,
                'status'
            )})`
        });
    }

// 6
private buildFilterFromArray(values: string[], prop: string) {
        return values?.map(e => `${prop} = '${e}'`).join(` or `) ?? '' undefined;
}
```

5. Внесите изменения в файл **app.ts**, используя следующий фрагмент кода

```
// 1
import { EventFilterBuilder as EventFilterBuilderNext } from './step2';

// 2
const filterResult = new EventFilterBuilderNext()
        .setCategoryFilter(orderFilter.category)
        .setStatusFilter(orderFilter.status)
        .setOpenOnlyFilter(true)
        .build();
    console.log(filterResult);


// No problem with mutation of filterResult
const openOnlyBuilder = new EventFilterBuilderNext().setOpenOnlyFilter(true);
const statusBuilder = openOnlyBuilder.setStatusFilter(orderFilter.status);
console.log('openOnlyBuilder result:', openOnlyBuilder.build()); // (status =
'apporved' or status = 'paid') and (deleted = 0)
console.log('statusBuilder result:', statusBuilder.build()); //  (status =
'apporved' or status = 'paid') and (deleted = 0)
```

6. Запустите проект, посмотрите результат работы приложения в консоли браузера.

Задание 3. Добавление строгой типизации

1. Создайте файл **src/step3/index.ts** и скопируйте в него содержимое файла **src/step2/index.ts**
2. Внесите изменения в класс **EventFilterBuilder** в файле **src/step3/index.ts** используя следующий фрагмент кода:

```
// 1
export type FilterType = {
    category: string;
    status: string | string[];
    openOnly: boolean;
};

// 2
export class EventFilterBuilder<T extends FilterType>

// 3
private filterResult: { readonly [key: string]: string } = {};
private filterResult: Record<keyof T, string> | undefined;

// 4
constructor(current = {})
constructor(current?: Record<keyof T, string>)

// 5
static #mergeObjects<TObject, TKey extends keyof TObject>(
        originalObject: TObject,
        changes: Pick<TObject, TKey>
    ): TObject {
        return Object.assign({}, originalObject, changes);
}

// 6
setCategoryFilter(category: string): EventFilterBuilder {
        return new EventFilterBuilder({
            ...this.filterResult,
            category: category ? `(category = '${category}')` : undefined,
        });
}
setCategoryFilter(category: string) {
        return new EventFilterBuilder(
            EventFilterBuilder.#mergeObjects(
                this.filterResult,
                { category: category ? `(category = '${category}')` : undefined }
            )
        );
}

// 7
setOpenOnlyFilter(openOnly: boolean): EventFilterBuilder {
        return new EventFilterBuilder({
```

```
            ...this.filterResult,
            openOnlyFilter: openOnly ? `(deleted = 0)` : undefined,
        });
}
setOpenOnlyFilter(openOnly: boolean) {
        return new EventFilterBuilder(
            EventFilterBuilder.#mergeObjects(
                this.filterResult,
                { openOnly: openOnly ? `(deleted eq 0)` : undefined }
            )
        );
}

// 8
setStatusFilter(status: string | string[]): EventFilterBuilder {
        const statusList = Array.isArray(status) ? status : [status];

        return new EventFilterBuilder({
            ...this.filterResult,
            statusFilter: `(${this.buildFilterFromArray(
                statusList,
                'status'
            )})`
        });
}
setStatusFilter(status: string | string[]) {
        const statusList = Array.isArray(status) ? status : [status];

        return new EventFilterBuilder(
            EventFilterBuilder.#mergeObjects(
                this.filterResult,
                { status: status ? `(${this.buildFilterFromArray(statusList,
'status')})` : undefined }
            )
        );
    }

// 9
build() {
        const filters = Object.values(this.filterResult!).filter(Boolean);
        return filters?.length > 0 ? filters.join(` and `) : '';
}

// 10
private buildFilterFromArray(values: string[], prop: string keyof T) {
        return values?.map(e => `${String(prop)} = '${e}'`).join(` or `) ??
undefined '';
    }
```

3. Внесите изменения в файл **app.ts**, используя следующий фрагмент кода

```
// 1
import { EventFilterBuilder as EventFilterBuilderImproved } from './step3';
```

```
// 2
const filterResult = new EventFilterBuilderImproved()
        .setCategoryFilter(orderFilter.category)
        .setStatusFilter(orderFilter.status)
        .setOpenOnlyFilter(true)
        .build();
    console.log(filterResult);


    const openOnlyBuilder = new
EventFilterBuilderImproved().setOpenOnlyFilter(true);
    const statusBuilder = openOnlyBuilder.setStatusFilter(orderFilter.status);
    console.log('openOnlyBuilder result:', openOnlyBuilder.build()); // (status =
'apporved' or status = 'paid') and (deleted = 0)
    console.log('statusBuilder result:', statusBuilder.build()); //  (status =
'apporved' or status = 'paid') and (deleted = 0)
```

4. Запустите проект, посмотрите результат работы приложения в консоли браузера.