Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота 8
З дисципліни: «Теорія розробки програмного забезпечення»
Web-browser
Visitor

Виконав:
студент групи ІА-14
Калінін Я.В

Київ 2023

Мета: Реалізувати паттерн visitor в проєкті на тему web-browser

Відвідувач — це поведінковий патерн проектування, що дає змогу додавати до програми нові операції, не змінюючи класи об’єктів, над якими ці операції можуть виконуватися.

Патерн Відвідувач пропонує розмістити нову поведінку в окремому класі, замість того, щоб множити її відразу в декількох класах. Об’єкти, з якими повинна бути пов’язана поведінка, не виконуватимуть її самостійно. Замість цього ви будете передавати ці об’єкти до методів відвідувача.

Код поведінки, імовірно, повинен відрізнятися для об’єктів різних класів, тому й методів у відвідувача повинно бути декілька. Назви та принцип дії цих методів будуть подібними, а основна відмінність торкатиметься типу, що приймається в параметрах об’єкта,

Патерн Відвідувач, використовує механізм подвійної диспетчеризації. Замість того, щоб самим шукати потрібний метод, ми можемо доручити це об’єктам, які передаємо в параметрах відвідувачеві, а вони вже самостійно викличуть правильний метод відвідувача.

Код:

```java
/**
 * Відображає структуру HTML поточної веб-сторінки.
 * Отримує повний HTML поточної сторінки та відображає його в WebEngine.
 */
2 usages  Qwertua *
public void displayHtmlStructureOnPage() {
    String htmlContent = (String) engine.executeScript( s: "document.documentElement.outerHTML");
    ContentVisitor visitor = new DisplayContentVisitor(engine);
    visitor.visitHtmlContent(htmlContent);
    closeButton.setDisable(false);
}
```

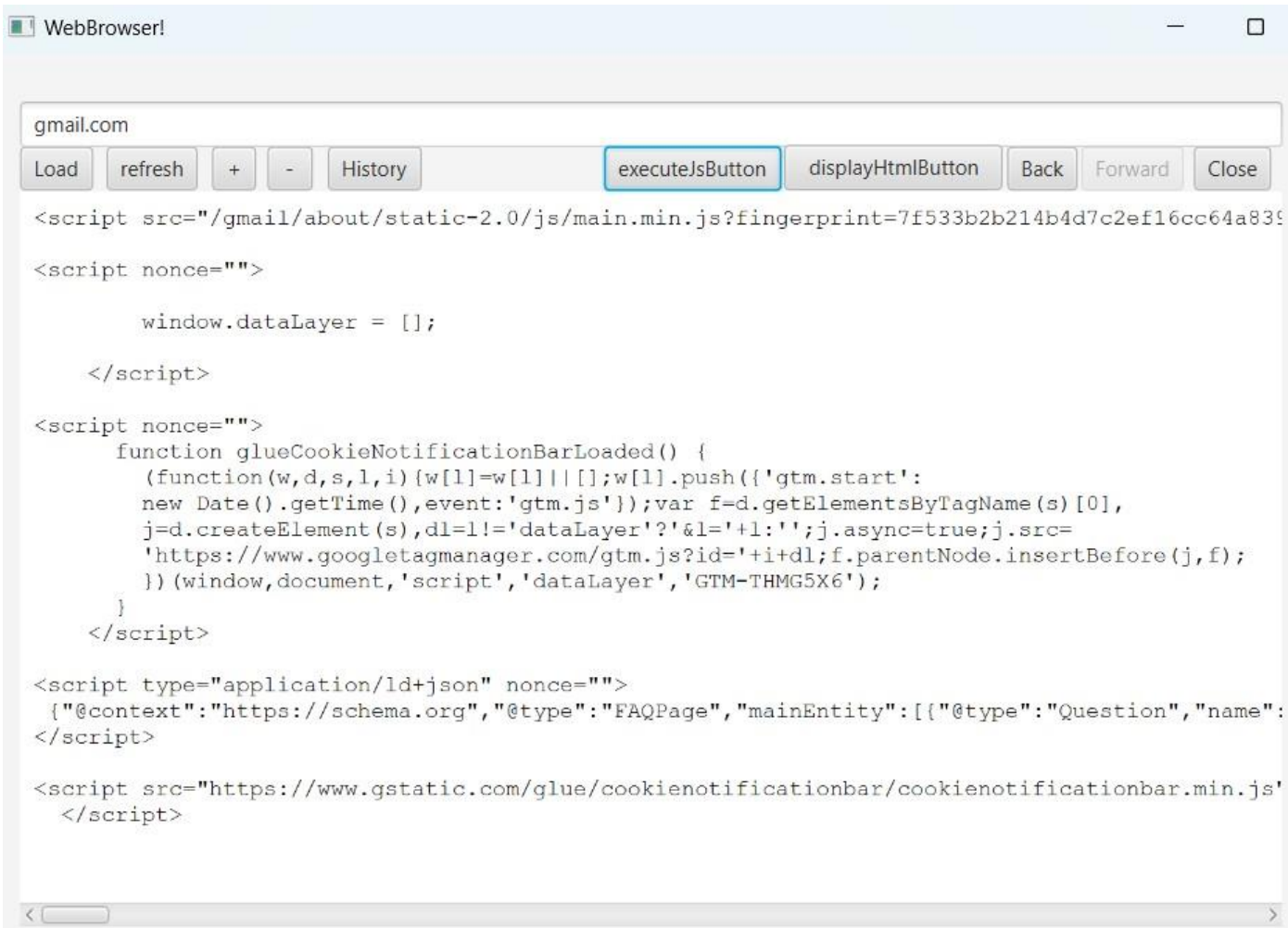
```java
/**
 * Виконує JavaScript на поточній веб-сторінці та відображає вихідні дані.
 * Збирає та відображає зміст усіх тегів <script> на поточній сторінці.
 */
2 usages  Qwertua *
public void executeJavaScript() {
    String script = """
        var scriptElements = document.getElementsByTagName('script');
        var jsContent = '';
        for (var i = 0; i < scriptElements.length; i++) {
            jsContent += scriptElements[i].outerHTML + '\\n\\n';
        }
        jsContent;
        """;
    String jsContent = (String) engine.executeScript(script);
    ContentVisitor visitor = new DisplayContentVisitor(engine);
    visitor.visitJsContent(jsContent);
    closeButton.setDisable(false);
}
```

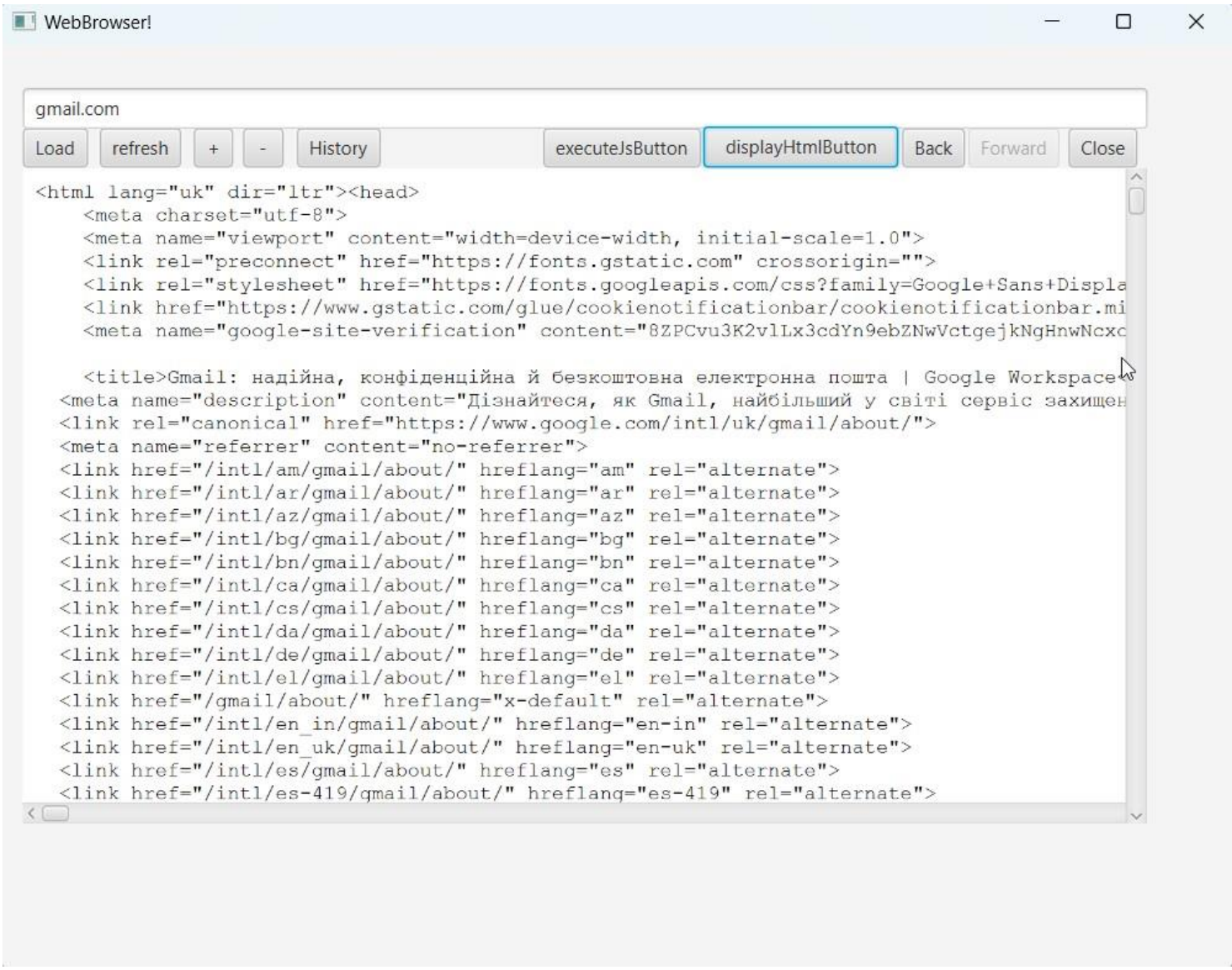
```java
package com.example.demo;

/**
 * Інтерфейс Visitor для обробки різних типів веб-контенту.
 */
1 implementation
public interface ContentVisitor {

    /**
     * Відвідує та обробляє HTML контент.
     *
     * @param htmlContent HTML контент для обробки.
     */
    1 usage   1 implementation
    void visitHtmlContent(String htmlContent);

    /**
     * Відвідує та обробляє JavaScript контент.
     *
     * @param jsContent JavaScript контент для обробки.
     */
    1 usage   1 implementation
    void visitJsContent(String jsContent);
}
```

```java
package com.example.demo;

import javafx.application.Platform;
import javafx.scene.web.WebEngine;

/**
 * Відвідувач для відображення HTML і JavaScript контенту.
 * Використовується для відображення контенту в WebEngine.
 */
public class DisplayContentVisitor implements ContentVisitor {
    3 usages
    private WebEngine engine;

    /**
     * Конструктор для DisplayContentVisitor.
     *
     * @param engine WebEngine для відображення контенту.
     */
    2 usages
    public DisplayContentVisitor(WebEngine engine) {
        this.engine = engine;
    }
```

```java
    /**
     * Відображає HTML контент у WebEngine.
     *
     * @param htmlContent HTML контент для відображення.
     */
    1 usage
    @Override
    public void visitHtmlContent(String htmlContent) {
        String formattedHtml = "<html><body><pre>" + escapeHtml(htmlContent) + "</pre></body></html>";
        Platform.runLater(() -> engine.loadContent(formattedHtml));
    }

    /**
     * Відображає JavaScript контент у WebEngine.
     *
     * @param jsContent JavaScript контент для відображення.
     */
    1 usage
    @Override
    public void visitJsContent(String jsContent) {
        String formattedJs = "<html><body><pre>" + escapeHtml(jsContent) + "</pre></body></html>";
        Platform.runLater(() -> engine.loadContent(formattedJs));
    }
```

```java
    /**
     * Допоміжний метод для екранування HTML.
     *
     * @param html Необроблений HTML контент.
     * @return Екранований HTML контент.
     */
    2 usages
    private String escapeHtml(String html) {
        return html
                .replace( target: "&",  replacement: "&amp;")
                .replace( target: "<",  replacement: "&lt;")
                .replace( target: ">",  replacement: "&gt;")
                .replace( target: "\"",  replacement: "&quot;")
                .replace( target: "'",  replacement: "&#39;");
    }
}
```


WebBrowser!
gmail.com
Load
refresh
+
-
History
executeJsButton
displayHtmlButton
Back
Forward
Close
<script src="/gmail/about/static-2.0/js/main.min.js?fingerprint=7f533b2b214b4d7c2ef16cc64a83
<script nonce="">
window.dataLayer = [];
</script>
<script nonce="">
function glueCookieNotificationBarLoaded() {
(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-THMG5X6');
}
</script>
<script type="application/ld+json" nonce="">
{"@context":"https://schema.org","@type":"FAQPage","mainEntity":[{"@type":"Question","name":
</script>
<script src="https://www.gstatic.com/glue/cookienotificationbar/cookienotificationbar.min.js'
</script>



Також за допомогою кнопки «Close» ми можемо виходити з режиму перегляду html css js структури документа ;

Контрольні запитання:

1. Які переваги шаблону «Відвідувач»?

Переваги шаблону "Відвідувач" включають:

Він дозволяє додавати нові операції до існуючих класів без зміни цих класів.

Відділяє алгоритми від об'єктів, на яких вони працюють.

Забезпечує зручний спосіб виконання операцій на групі об'єктів різних класів.

2. Які недоліки шаблону «Відвідувач»?

Недоліки шаблону "Відвідувач" включають:

Він може ускладнювати код, якщо ієрархія об'єктів часто змінюється.

Може порушувати інкапсуляцію об'єктів, оскільки відвідувач повинен мати доступ до внутрішнього стану цих об'єктів.

Ускладнює підтримку коду, особливо коли додаються нові класи до ієрархії, оскільки це вимагає оновлення усіх відвідувачів.