אפיון ומדריך לביצוע

מערכת: פרקטיקוד
קורס: Html Serializer כלי - Experience 2
ספר: אפיון ומדריך לביצוע

הודפס על-ידי: תמר גורביץ
תאריך: 12:50 ,8/01/2024

תוכן עניינים

תיאור הפרויקט

בפרויקט זה נפתח כלי לטיפול ועיבוד של Html.

ניתן להשתמש בכלי כזה לצרכים שונים, לדוגמא כדי לממש Crawler.

Crawler (או Scrapper) הוא מנגנון שקורא אתרי אינטרנט ומנתח את ה-Html שלהם כדי לחלץ ממנו את המידע הרצוי. למעשה, זה בדיוק מה שעושה מנוע החיפוש של גוגל (ואחרים). המנוע סורק את האינטרנט ומנתח את ה-Html של כל האתרים במטרה לאנדקס את המידע כך שיוכל לאחזר אותו בהתאם לשאילתות החיפוש שהמשתמשים מבקשים.

דוגמאות נוספות לשימוש ב-Crawling:

ניתוח אתרי אינטרנט כדי לגלות באיזה טכנולוגיות הם כתובים ובאיזה ספריות קוד הם משתמשים.

שליפת נתונים מאתרי קניות או יד שניה כדי להציג באתר אחר.

ועוד ועוד.

בפרויקט זה נפתח את הקוד התשתיתי שבהמשך נוכל להשתמש בו כדי לפתח Crawler משלנו.

הכלי שנפתח מורכב משני חלקים: Html Serializer ו-Html Query.

Html Serializer

סריאליזציה - תהליך של המרת נתונים מפורמט מסוים לאוביקטים של שפת תכנות.

עלייך לפתח שירות שניגש לכתובת של דף אינטרנט, קורא את ה-Html שחוזר ממנה וממיר את ה-Html לאוביקטים של #C.

שלבי הפיתוח מפורטים בפרקים הבאים.

קריאה לדף אינטרנט

השתמשי באוביקט HttpClient כדי לבצע קריאה לכתובת אינטרנט.

דוגמת קוד

```
1 reference
public async Task<string> Load(string url)
{
    HttpClient client = new HttpClient();
    var response = await client.GetAsync(url);
    var html = await response.Content.ReadAsStringAsync();
    return html;
}
```

לאחר הקריאה לכתובת האינטרנט, קיבלת מחרוזת שמכילה את כל ה-Html.

פירוק לפי תגיות

פרקי את המחרוזת לחלקים.

השתמשי ב-Regular Expression שמחפש מבנה של תגית html (מתחילה ב-> ומסתיימת ב-<)

נקי את כל המחרוזות הריקות, ירידות השורה והרווחים המיותרים.

ראי סרטון הדרכה בשם Regular Expressions in .NET

בסיום הפירוק יש לך אוסף של מחרוזות שמהן תוכלי לבנות את האוביקטים.

מחלקת HtmlElement

צרי מחלקה שמייצגת תגית של Html.

המחלקה תכיל את המאפיינים הבאים:

Id

Name

Attributes (רשימה)

Classes (רשימה)

InnerHtml

בנוסף, המחלקה תכיל מאפיינים של Parent ו-Children שיאפשרו לך ליצור עץ של אוביקטים.

מחלקת HtmlHelper

פתחי מחלקת עזר שתספק לך את רשימת תגיות ה-HTML בצורה נוחה. (נדרש שימוש בהמשך)

מצורפים קבצי JSON שמכילים רשימת תגיות. אחד מכיל את כל התגיות הקיימת ב-HTML והשני מכיל את התגיות שלא דורשות סגירה של התגית.

מחלקת HtmlHelper תכיל שני מאפיינים מסוג מערך של string.

ב-constructor טעני את הנתונים מקבצי ה-JSON למערכים.

1. קראי את תוכן הקובץ באמצעות מחלקת File.
   ראי סרטון הדרכה בשם **How to copy file to output directory**
2. המירי את הנתונים למערך באמצעות הפונקציה Deserialize של מחלקת JsonSerializer (יש להוסיף using ל-System.Text.Json)

Singleton

מחלקת HtmlHelper היא מקרה קלאסי שמתאים לתבנית עיצוב של Singleton.

Singleton נועד למקרים שבהם אנחנו מעונינים להגביל את מספר המופעים של המחלקה לאחד בלבד.

במקרה של HtmlHelper מובן שאין טעם וחבל ליצור מופע חדש בכל פעם, מה שיגרור טעינה מחדש של קבצי ה-JSON ובזבוז משאבים לחינם.

אופן המימוש של Singleton פשוט מאוד.

📺 **ראי סרטון הדרכה בשם Singleton in .NET**

בניית העץ

עקרונית, ניתן לרוץ על רשימת המחרוזות בריקורסיה כדי לבנות את העץ. אך למעשה, קיימת הגבלה של כמות הקריאות לפונקציה שה-Stack יכול להחזיק. הגבלה זו עלולה לגרום לשגיאת Stack Over Flow במקרה שה-Html גדול.

צרי את אוביקט השורש וכן משתנה נוסף שיחזיק את האלמנט הנוכחי בכל איטרציה.

רוצי בלולאה על רשימת המחרוזות ובכל איטרציה בצעי את התהליך הבא:

חתכי את המילה הראשונה במחרוזת, ובדקי:

1. אם היא "/html" סימן שהגעת לסוף ה-html.

2. אם היא מתחילה ב-"/" סימן שמדובר בתגית סוגרת - עלי לרמה הקודמת בעץ (כלומר, שימי באלמנט הנוכחי את האבא)

3. אם היא שם של תגית מתגיות ה-Html צרי אוביקט חדש והוסיפי אותו לרשימת ה-Children של האלמנט הנוכחי.
(מצורף קובץ JSON שמכיל את רשימת התגיות ב-HTML, לפי הקישור הזה:
https://developer.mozilla.org/en-US/docs/Web/HTML/Element)
פרקי את המשך המחרוזת (ללא המילה הראשונה) וצרי את רשימת ה-Attributes.
כדי לפרק את המשך המחרוזת בצורה נכונה עלייך להשתמש ב-Regular Expression.
ראי סרטון הדרכה בשם Regular Expressions in .NET
אם יש Attribute בשם class פרקי אותו לחלקים לפי רווח (כיון שניתן להגדיר כמה classים על אלמנט Html) ועדכני את המאפיין Classes בהתאם.
שימי במאפיין Name את שם התגית וב-Id את המזהה (מה-attributes, אם יש)
כמו כן עדכני את ה-Parent בהתאם.
בנוסף, בדקי אם התגית סוגרת את עצמה (self-closing) -
אם המחרוזת מסתיימת ב-"/" או שהתגית נמצאת ברשימת התגיות שלא דורשות תגית סגירה.
(מצורף קובץ JSON שמכיל את רשימת התגיות שלא דורשות סגירה, לפי הקישור הזה https://www.scaler.com/topics/self-closing-tags-in-html/
)
במקרה כזה השאירי את האלמנט הנוכחי כמות שהוא. אך אם לא, סימן שיש תגית סגירה נפרדת ויתכן שיש תוכן פנימי לתגית הנוכחית (תגיות ילדים או InnerHtml) ולכן שימי באלמנט הנוכחי את האוביקט החדש שיצרת.

4. אם היא לא אף אחת מהאפשרויות הנ"ל, סימן שמדובר בטקסט פנימי של האלמנט. עדכני את המאפיין InnerHtml של האלמנט הנוכחי בערך של המחרוזת.

כעת יש בידייך עץ של אוביקטי HtmlElement 🎉

Html Query

פתחי שירות לתשאול עץ של אוביקטי HtmlElement וחיפוש אלמנטים לפי CSS Selector.

ממשי חיפוש בסיסי לפי שם תגית, id ושם class.

כללי השאילתה הם כדלהלן:

לחיפוש לפי תגית יש לשרשר את שם התגית, לדוגמא:

"div"

לחיפוש לפי id יש לשרשר את ה-id עם תחילית של #, לדוגמא:

"#mydiv"

לחיפוש לפי class יש לשרשר את שם ה-class עם תחילית של . (נקודה), לדוגמא:

".class-name"

שרשור של כמה סלקטורים ללא רווח מציין חיפוש אלמנטים שמקיימים את כל התנאים, לדוגמא:

"div#mydiv.class-name"

השאילתה לעיל מחפשת אלמנטים מסוג div עם id שערכו mydiv ושיש להם class בשם class-name.

רווח בין חלקי ה-selector מציין חיפוש בדורות הבאים של האלמנט (לעומק), לדוגמא:

"div #mydiv .class-name"

השאילתה לעיל מחפשת אלמנטים שיש להם class בשם class-name ויש להם אבא (לא חובה ישיר) עם id שערכו mydiv, כשלאבא הזה עצמו יש אבא מסוג div.

שלבי הפיתוח מפורטים בפרקים הבאים.

מחלקת Selector

צרי מחלקה שמייצגת Selector.

המחלקה תכיל את המאפיינים הבאים:

TagName

Id

Classes (רשימה)

בנוסף המחלקה תכיל מאפיינים של Parent ו-Child שיאפשרו לך לבנות היררכיה של סלקטורים.

(במקרה הזה כל אבא מכיל בן אחד בלבד - עץ בינארי מנוון)

כתבי פונקציה סטטית במחלקת Selector שממירה מחרוזת של שאילתה לאוביקט Selector.

פרקי את המחרוזת לחלקים לפי הרווחים, כך שתקבלי אוסף של מחרוזות שכל אחת מייצגת רמה נוספת בשאילתה.

צרי את אוביקט השורש וכן משתנה נוסף שיחזיק את הסלקטור הנוכחי בכל איטרציה.

רוצי בלולאה על רשימת המחרוזות ובכל איטרציה בצעי את התהליך הבא:

1. פרקי את המחרוזת לחלקים לפי המפרידים # ו-. (נקודה)
2. עדכני את מאפייני הסלקטור הנוכחי בהתאם לחלקי המחרוזת.
   אם יש חלק במחרוזת שלא מתחיל ב-# או ב-. יש לבדוק אם הוא שם תקין של תגית HTML ורק אם כן, לשים אותו במאפיין TagName.
3. צרי אוביקט Selector חדש, הוסיפי אותו כבן של הסלקטור הנוכחי ועדכני את הסלקטור הנוכחי שיצביע עליו.

פונקציות ב-HtmlElement

ממשי שתי פונקציות לריצה על העץ במחלקת HtmlElement.

פונקצית Descendants

הפונקציה רצה על כל העץ מהאלמנט שעליו היא מופעלת ומטה, ומחזירה רשימה שטוחה של כל צאצאי האלמנט (מכל הדורות).

כיון שהעץ עשוי להיות גדול, ריקורסיה עלולה לתקוע את ה-Stack, לכן ממשי את הפונקציה באמצעות תור - Queue.

צרי Queue של אוביקטים מסוג HtmlElement ודחפי לו את האלמנט הנוכחי (this).

רוצי עם לולאת While כל זמן שהתור לא ריק, ובכל איטרציה בצעי את התהליך הבא:

1. שלפי את האוביקט שבתחילת התור.
2. הוסיפי אותו לרשימה באמצעות yield return.
   ראי סרטון הדרכה בשם [IEnumerable & yield return]
3. רוצי בלולאה על הבנים של האוביקט (אם ישנם) והוסיפי אותם לתור.

פונקצית Ancestors

הפונקציה רצה על כל כל העץ מהאלמנט שעליו היא מופעלת ומעלה, ומחזירה רשימה שטוחה של כל אבות האלמנט (מכל הדורות).

ממשי את הפונקציה.

שימי לב, מימוש הפונקציה פשוט יותר מפונקצית Descendants ולא דורש ריקורסיה או תור.

מציאת אלמנטים בעץ לפי סלקטור

ממשי פונקצית הרחבה למחלקת HtmlElement שתקבל Selector ותחזיר רשימת אלמנטים שעונים לקריטריונים של הסלקטור.

הפונקציה תשתמש בריקורסיה כדי לרוץ על עץ האלמנטים ועץ הסלקטור.

הפונקציה הריקורסיבית תקבל HtmlElement ו-Selector וכן אוסף של HtmlElement שיחזיק את התוצאה של החיפוש.

השתמשי בפונקציית Descendants כדי לקבל את רשימת כל הצאצאים של האלמנט הנוכחי.

חפשי ברשימת הצאצאים את כל אלו שעונים לקריטריונים של הסלקטור הנוכחי.

הפעילי את הריקורסיה על הרשימה המסוננת עם הסלקטור הבא (הבן של הנוכחי).

אם הגעת לסלקטור האחרון (תנאי עצירה), הוסיפי את הרשימה המסוננת לאוסף שמחזיק את תוצאת החיפוש.

מניעת כפילויות באמצעות HashSet

אלמנט בעץ יכול לענות לתנאי ה-selector יותר מפעם אחת. לדוגמא אם בעץ יש היררכיה כזו:

```
<div>
    <div>
        <p class="class-name"></p>
    </div>
</div>
```

אם נרשום selector כזה: "div p.class-name", האלמנט p יענה פעמיים לתנאי, פעם אחת בגלל ה-div הראשון (הסבא), ופעם נוספת בגלל ה-div השני (האבא)

ניתן למנוע את הכפילות בתוצאת השאילתה בשתי דרכים:

1. הדרך הפרימיטיבית - הוסיפי מאפיין דגל בשם Visited במחלקת HtmlElement והדליקי אותו בפעם הראשונה שהאלמנט יענה לתנאי. בפעם הבאה שהאלמנט יענה לתנאי, לא תצרפי אותו לתוצאה כיון שהדגל כבר דלוק.
   כאמור זו דרך פרימיטיבית ומסורבלת. לא מומלץ להשתמש בה.

הדרך המומלצת - השתמשי במחלקת HashSet עבור תוצאת החיפוש.
HashSet בנוי כך שכאשר מוסיפים לו אוביקט הוא בודק אם האוביקט כבר קיים, אם כן, הוא פשוט לא מוסיף אותו, כדי למנוע כפילות.
ראי סרטון הדרכה בשם HashSet in .NET

מעקב ובדיקה במהלך הפיתוח

כדי לבדוק אם רשימת האלמנטים שהפונקציה מחזירה לנו אכן נכונה ותקינה ניתן לבדוק בדפדפן על האתר עצמו.

פתחי את ה-console ורשמי את הסלקטור הרצוי כך:

```
$$("div .class-name")
```

תקבלי את רשימת האלמנטים שעונים לקריטריון ותוכלי להשוות את התוצאה.

❗ הערה חשובה

לעיתים יהיה הבדל בתוצאה שלא נובע מבעיה בפונקציה. פעמים רבות יש קוד javascript שמוסיף אלמנטים לדף מעבר ל-html הבסיסי שחזר מהקריאה לכתובת האתר.

אם הסלקטור בקונסול מחזיר יותר אלמנטים מהפונקציה, תוכלי לבדוק מול ה-html המקורי בלחיצה ימנית על האתר ובחירה בתפריט 'הצגת מקור הדף'.

📺 **ראי סרטון הדרכה בשם** [How to validate our query]