Floraguard
Configuration Setup

This document was created as part of the project Harnessing Technology to End the Illegal Trade in Succulent Plants, conducted by Royal Botanic Gardens, Kew and the University of Southampton between 2022 and 2025, funded by the UK Government through the Illegal Wildlife Trade Challenge Fund.

Contents

1 Introduction

In this section, we will go over how to collect the necessary information to set up the parser configuration for both forums and marketplaces. This process involves identifying specific parts of a web page using your browser's developer tools.

Before continuing, we assume you either have a basic understanding of HTML or have read the accompanying ***HTML Primer*** included with this document.

To help you along, this guide includes screenshots from both Google Chrome and Microsoft Edge. Figures 1 and 2 show what the Inspect window looks like in each browser.

Note: The screenshots are shown in target website **(light mode)** and HTML inspection pane **(dark mode)** respectively to make it easier to tell the difference between them.

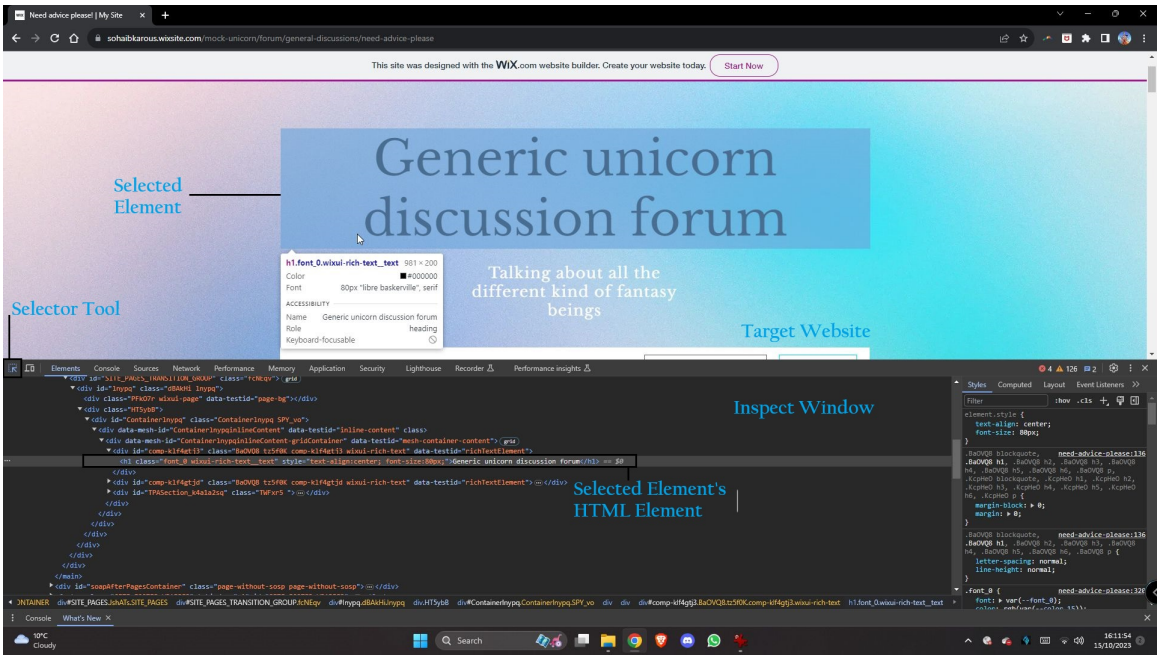

Figure 1: Inspecting HTML in **Google Chrome**

To open the Inspect window, follow these steps:

1. Navigate to the website you want to inspect in any modern browser.
2. **Right-click** on any part of the page.
3. Select **Inspect** or **Inspect Element** from the menu.

Shortcut: `Ctrl+Shift+I` (works in most browsers)

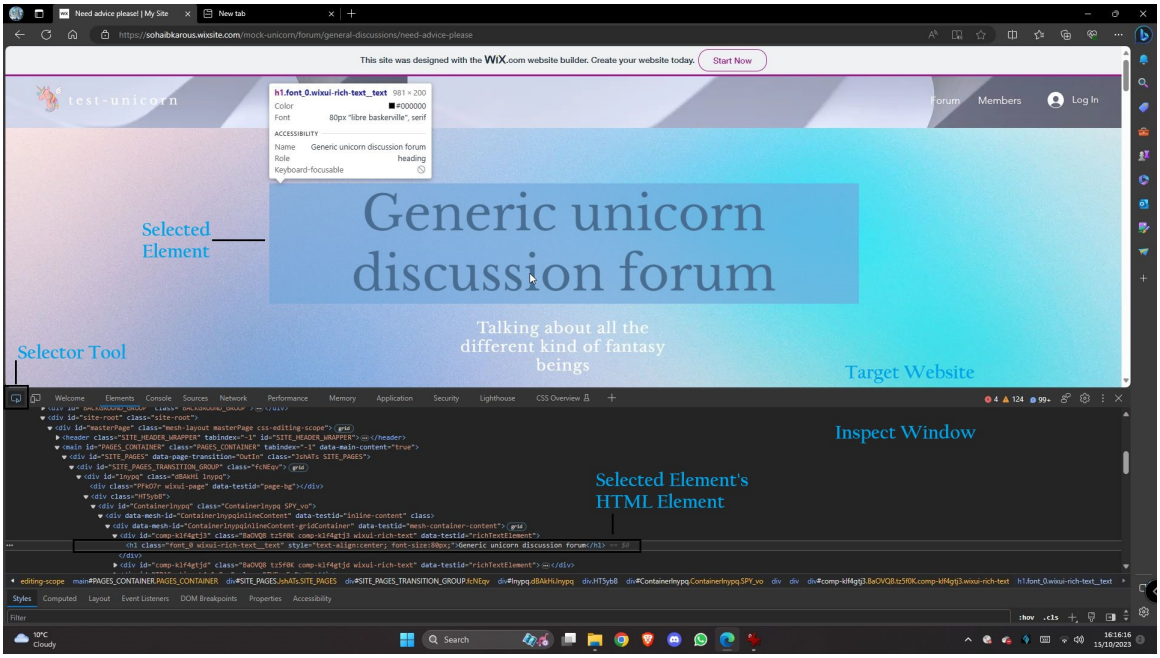

Figure 2: Inspecting HTML in **Microsoft Edge**

Once the Inspect window is open, we want to identify two key pieces of information for a webpage element:

- The **name of the HTML tag** (e.g., `div` ,`span` or `h1`)
- The **class name** (used to describe what kind of element it is)

We'll use two methods to find this information, described below.

1.1 Method 1: Hover Over the Element

This is the most straightforward method and works in most cases.

1. Click the **Selector Tool** (the icon that looks like a cursor or arrow) in the top-left corner of the Inspect window. See Figures 1 and 2.
2. Hover your mouse over the element you're interested in (e.g., a post title or username). A light blue box will highlight the area on the page.
3. When the correct area is highlighted, **left-click** on it.
4. The corresponding HTML code will now be shown and highlighted in the Inspect window.

From here, note the tag name (such as `div`) and the class name (e.g., `post-title`) listed in the HTML.

1.2 Method 2: Find the Parent (Superclass) Tag

Sometimes, the element you're trying to inspect doesn't have a class name — or selecting it directly doesn't give the full content. In these cases, we look slightly "above" it in the HTML structure to find its parent tag (also called the *superclass*).
Follow these steps:

1. First, follow Method 1 to inspect the element.
2. In the Inspect window, move your cursor up through the HTML structure to inspect the parent tags.
3. As you hover over each parent, you'll see the highlighted region change on the website. Stop when the entire section you care about is fully highlighted.
4. This parent tag is often a better choice — especially if it has a class name.

Using this method, you can find the most useful tag/class combination for use in your configuration.

2 Common Configuration Parameters

2.1 name

The `name` field now refers to the filename of your custom website configuration (excluding the file extension). This is how the system identifies and loads the correct configuration file.

- The value must **match exactly** the name of your configuration file located in `resources/custom_webpages/`.
- Do **not** include the `.json` extension — just use the filename.

Example:

- If your config file is named `mock_site.json`, then set `name` to:

```
"name": "mock_site"
```

2.2 type

This defines the **category** of the configuration.

- Use `forum` for discussion-based websites.
- Use `marketplace` for sites that list products or items for sale.

2.3 root_page_url

This is the main entry point — the homepage or root URL of the site you want to extract data from.

- For example: `https://example.com/forum` or `https://marketplace.org/shop`

3 Forum Configuration Setup


```
{
  "name" : "mocked",
  "type" : "forum",
  "root_page_url": "https://sohaibkarous.wixsite.com/mock-unicorn",
  "general_threads_page_url" : "https://sohaibkarous.wixsite.com/mock-unicorn/forum/",
  "general_thread_url" : "https://sohaibkarous.wixsite.com/mock-unicorn/forum/",
  "general_profile_url" : "https://sohaibkarous.wixsite.com/",
  "thread_name_regex" : {"name" : "div", "class_" : "NWUMmd"},
  "block_regex" : {"name" :  "div" , "class_" :  ["TD_VuS","sgEKh8 top-level-comment", "sC_Tkp9 wc-app-desktop", "sUxkE8_"]},
  "comment_regex" : {"name" : "span", "class_" :  ["d0767", "_2PHJq public-DraftStyleDefault-ltr"]},
  "profile_regex" : {"name" : "div", "class_" :  ["mzrlgz","r05Xnt","EcX0n5"]},
  "profile_name_regex" : {"name" : "span", "class_":  ["ZREX02 forum-text-color forum-link-hover-color", "s__1Rtiwz"]},
  "profile_link_regex" : {"name" : "a"},
  "date_regex" : {"name" :  "span", "class" : ["NGGLat TRnkW5 time-ago", "s__7oOHwa"]},
  "attributes_regex": {"username": {"name" :  "h2", "class_": "forum-title"}, "user_stats" : {"name" :  "dl", "class_" : "details" } }
}
```


Figure 3: Configuration Setup for Forums

Figure 3 shows the basic configuration setup for a mock site of type `forum`, which can be found at `"resources/custom_webpages/mock_site.json"`. In the following subsections, we explain how to fill out each part of the forum-style configuration. Common parameters are described in Section 2.

3.1 general_threads_page_url

This is the URL of the main forum page on the target website.

3.2 general_thread_url

This field defines the part of the URL that is common to all discussion threads. The crawler uses this as the starting point. Open several threads to identify the common portion.
Example:

- `https://sohaibkarous.wixsite.com/mock-unicorn/forum/general-discussions/wanted`
- `https://sohaibkarous.wixsite.com/mock-unicorn/forum/general-discussions/dora-the-unicorn`
- `https://sohaibkarous.wixsite.com/mock-unicorn/forum/finding-your-own-unicorns/unicorn-identification-books-for-sale`
- `https://sohaibkarous.wixsite.com/mock-unicorn/forum/dealing-with-unicorns/wild-species`

The common part for all URLs above is:

```
"general_thread_url": "https://sohaibkarous.wixsite.com/mock-unicorn/forum/"
```

3.3 general_profile_url

If forum user profiles are available, identify the part of the profile URLs that is shared across all profiles. Follow the same approach as in Section 3.2.

3.4 thread_name_regex

This identifies the HTML element containing the thread's title.

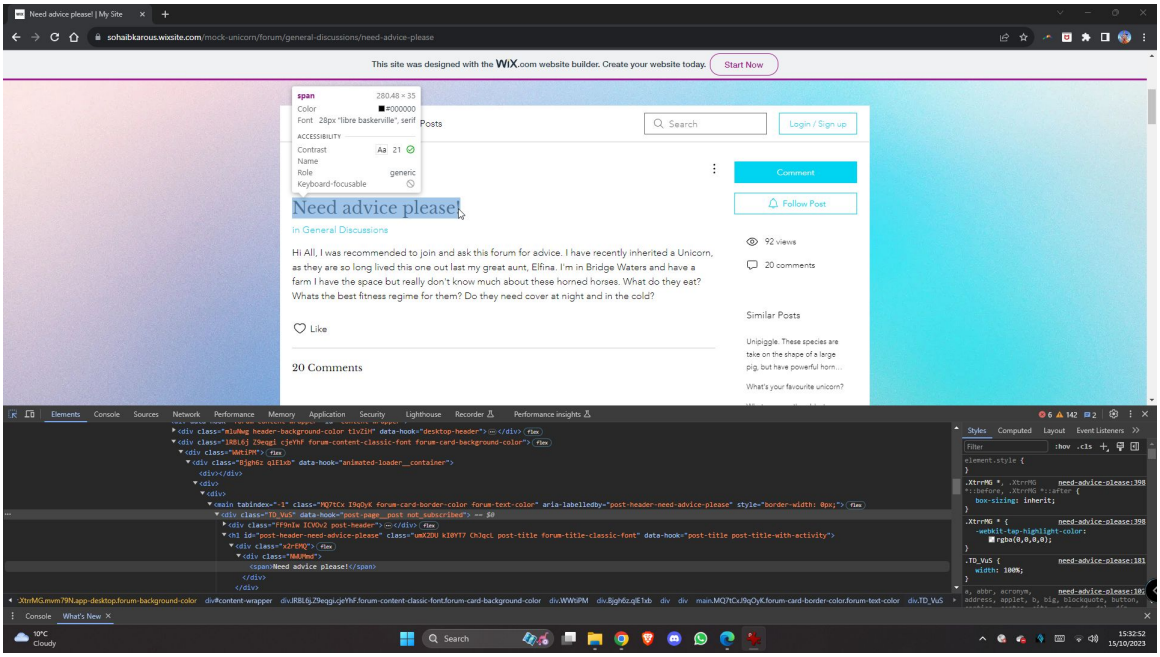

Figure 4: Thread title in Google Chrome

Using the selector tool (Figure 1), we hover over the thread title. The highlighted tag is:

```
<span>Need advice please!</span>
```

Since `<span>` does not have a class, we look at its parent (superclass):

```
<div class="NWUMmd">...</div>
```

We extract the following config:

```
"thread_name_regex": {
    "name": "div",
    "class_": "NWUMmd"
}
```

Repeat this with several threads to identify consistent class patterns.

3.5 block_regex

This field captures the main content block of a thread.

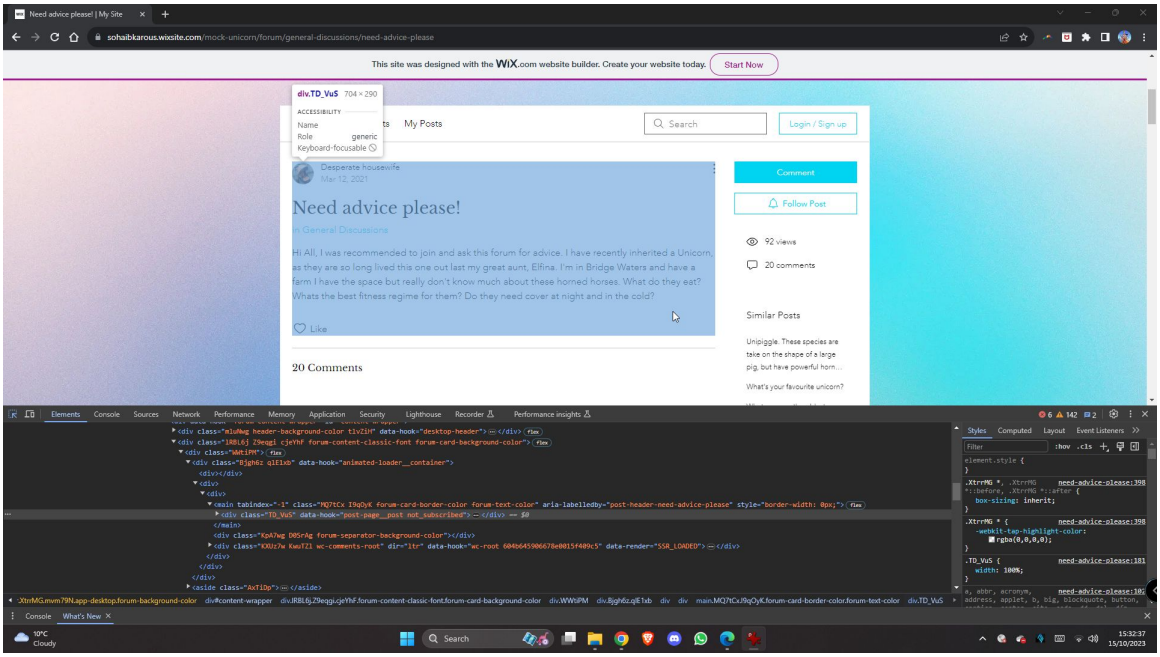

Figure 5: Thread block in Google Chrome

Hovering over the thread block highlights:

```
<div class="TD_VuS" data-hook="post-page__post not_subscribed">
```

Use this information:

```
"block_regex": {
    "name": "div",
    "class_": "TD_VuS"
}
```

3.6 comment_regex

This field extracts user comments or replies.

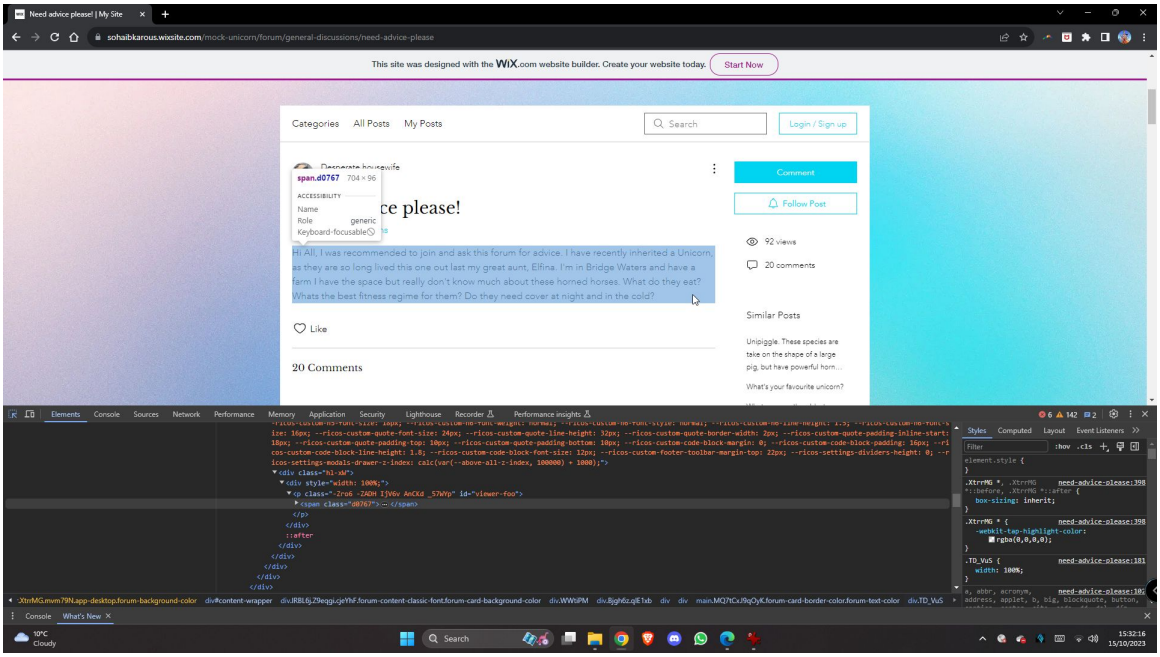

Figure 6: Comment block in Google Chrome

The tag highlighted is:

```
<span>"Hi All ...."</span>
```

Its parent tag:

```
<span class="d0767">...</span>
```

Results in:

```
"comment_regex": {
    "name": "span",
    "class_": "d0767"
}
```

3.6.1 Cases of Multiple Paragraphs

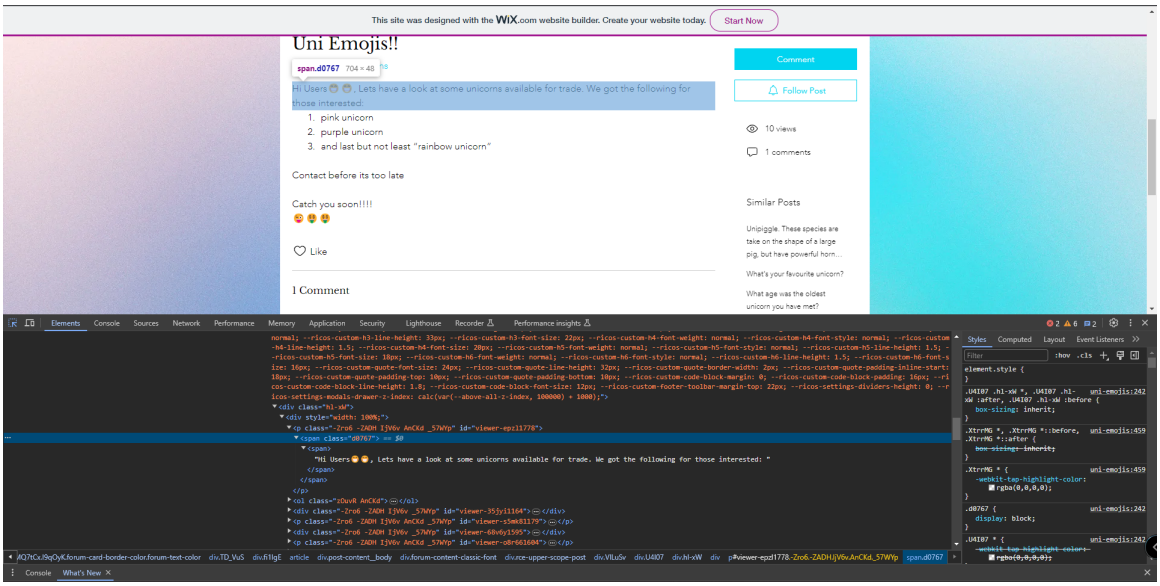

Figure 7: Comment with multiple spans

HTML sometimes splits paragraphs or bullet points into multiple `<span>` elements. In such cases, look for a common parent `<div>` element.

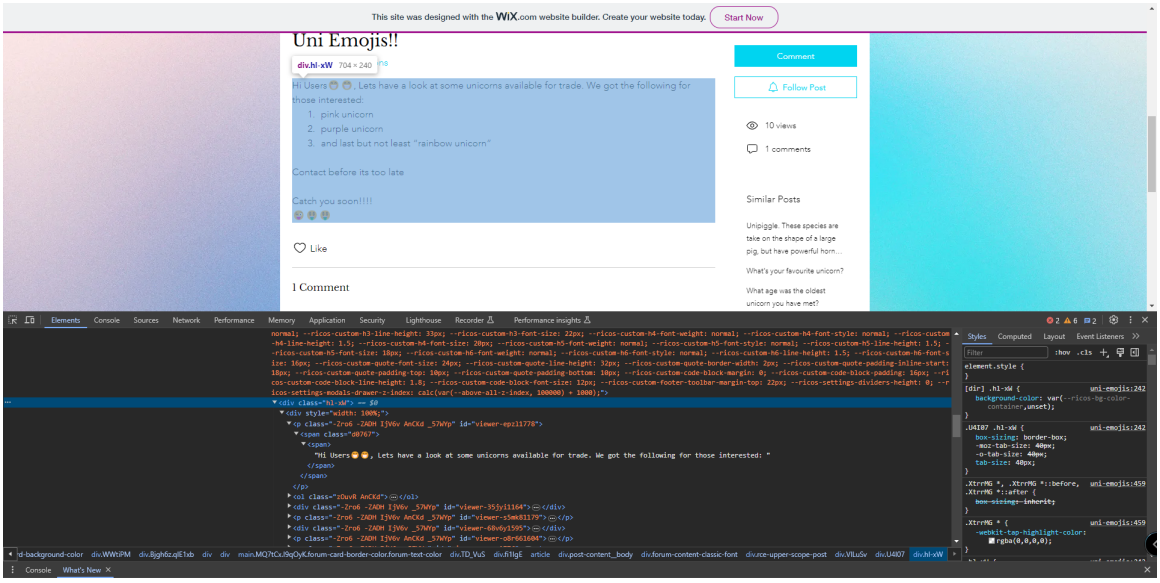

Figure 8: Parent `<div>` element containing comment spans

Extract the class from that parent `<div>`.

Example:

```
"comment_regex": {
    "name": "div",
    "class_": "h1-xW"
}
```

3.7 profile_regex

Identifies the HTML block containing the profile information.

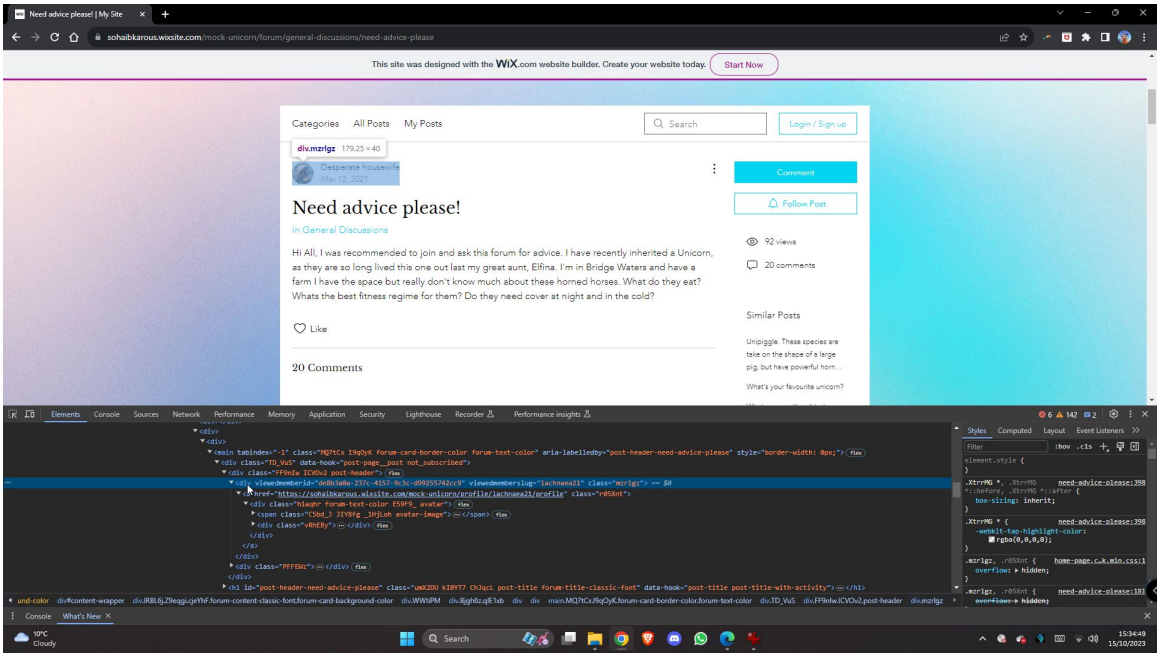

Figure 9: Profile section in Google Chrome

Initially, the element is:

```
<div class="h1a...">...</div>
```

However, this does not contain the link tag `<a href="...">`. So we look at its superclass.

Its superclass:

```
<div class="mzrlgz">...</div>
```

Use:

```
"profile_regex": {
    "name": "div",
    "class_": "mzrlgz"
}
```

3.8 profile_name_regex

This identifies the HTML tag containing the user's profile name.

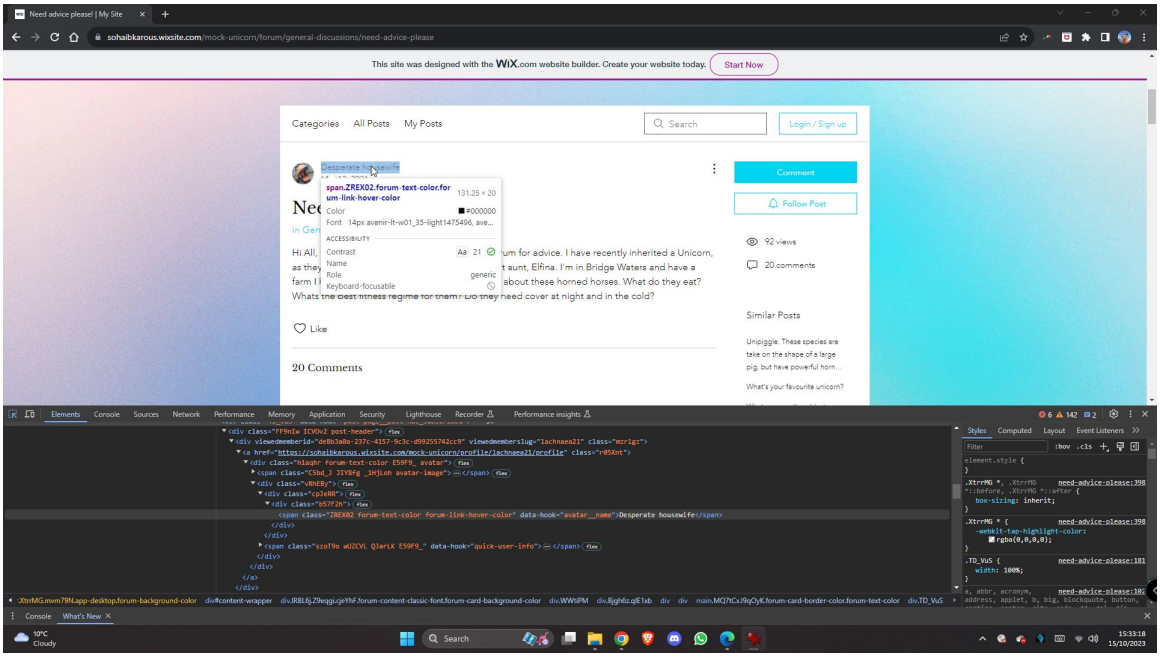

Figure 10: Profile name highlighted in Google Chrome

```
<span class="ZREX02 forum-text-color forum-link-hover-color">...</span>
```

However, in this case span contains definitions for color and links such as *"forum-text-color forum-link-hover-color"*. So we look at is superclass.

Its superclass:

```
<div class="b57F2h"> ... </div>
```

Use:

```
"profile_name_regex": {
    "name": "div",
    "class_": "b57F2h"
}
```

3.9 profile_link_regex

This captures the URL to the user's profile.

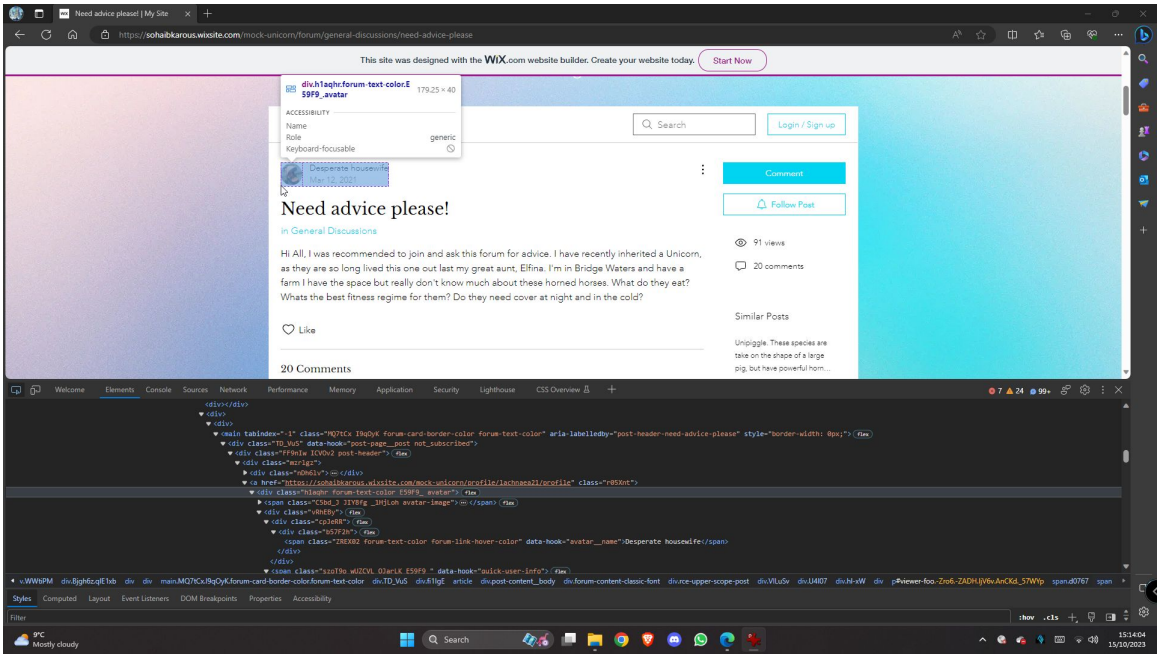

Figure 11: Profile Link superclass

Look for a tag like:

```
<a href="https://..." class="r05Xnt">...</a>
```

We only need the tag name:

```
"profile_link_regex": {
    "name": "a"
}
```

3.10 date_regex

This field identifies the date a thread was posted.

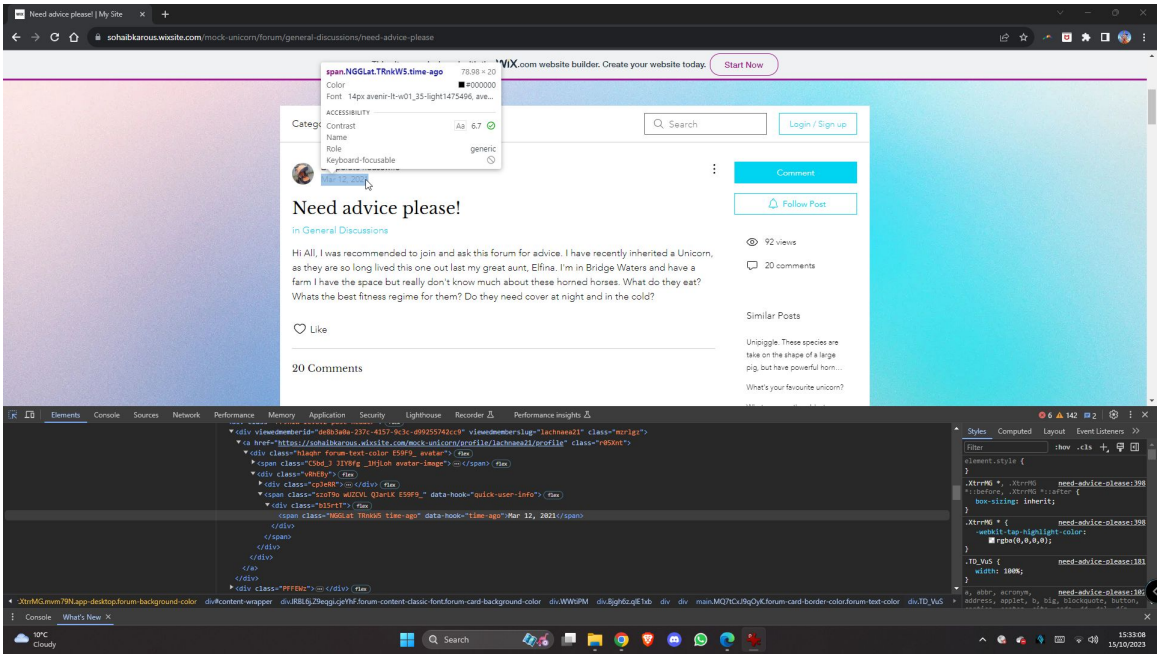

Figure 12: Date highlighted in Google Chrome

```
<span class="NGGLat TRnkW5 time-ago" data-hook="time-ago">Mar 12, 2021</span>
```

However, in this case span contains definitions for time caculation such as *"time-ago"*. So we look at is superclass.

Its superclass

```
<div class="b15rtT">...</div>
```

Use:

```
"date_regex": {
    "name": "div",
    "class_": "b15rtT"
}
```

3.11 attributes_regex

This optional field lets you extract any additional elements from the thread page.
Example:

```
"attributes_regex": {
    "post_id": {
        "name": "div",
        "class_": "post-container"
    },
    "user_status": {
        "name": "span",
        "class_": "badge"
    }
}
```

4 Marketplace Configuration Setup

Figure 13: Configuration Setup for Marketplaces

Figure 13 shows the basic configuration setup for a mock site of type `marketplace`, located at `"resources/custom_webpages/caps.json"`. The following subsections explain how to complete each part of a marketplace-style configuration. Common parameters are covered in Section 2.

4.1 general_items_url

This is the starting URL of the marketplace — typically a product list or category page.

4.2 general_item_url

This field defines the part of the URL that is common to all individual item listings. Refer to Section 3.2 for guidance on identifying common URL patterns.

4.3 sale_item_name_regex

This field identifies the HTML tag containing the product title.

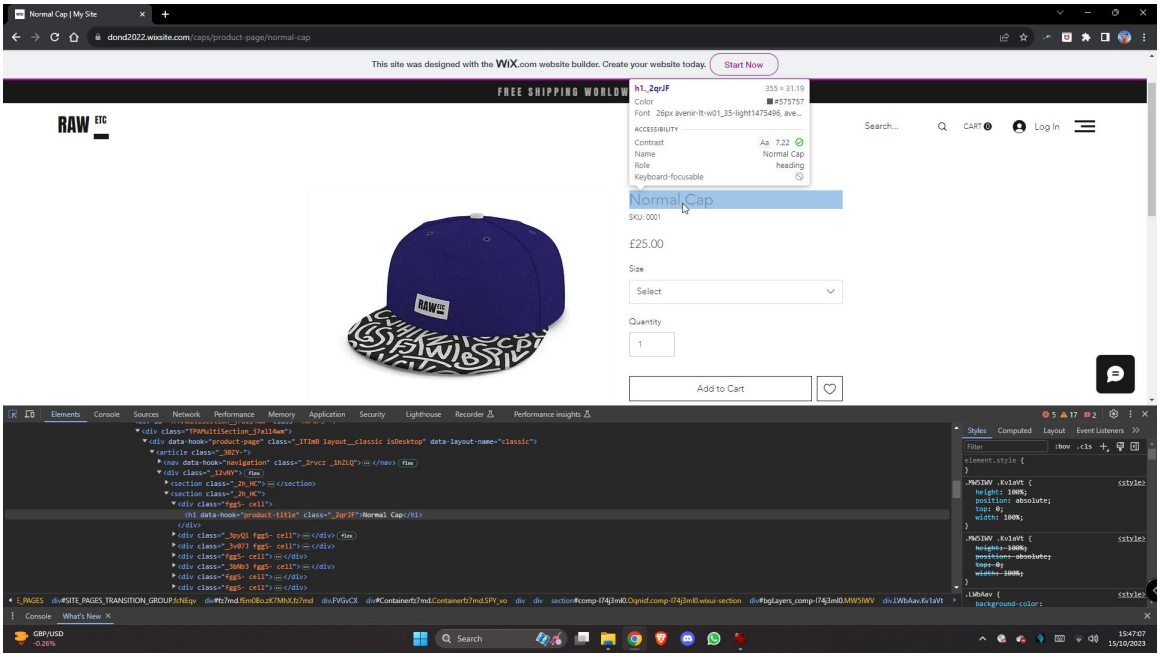

Figure 14: Product title in Google Chrome

Hovering over the item name reveals:

```
<h1 data-hook="..." class="_2qrJF">...</h1>
```

Use the following configuration:

```
"sale_item_name_regex": {
    "name": "h1",
    "class_": "_2qrJF"
}
```

Repeat this process with other listings to confirm consistency in tag and class.

4.4 sale_item_description_regex

This captures the product description block.

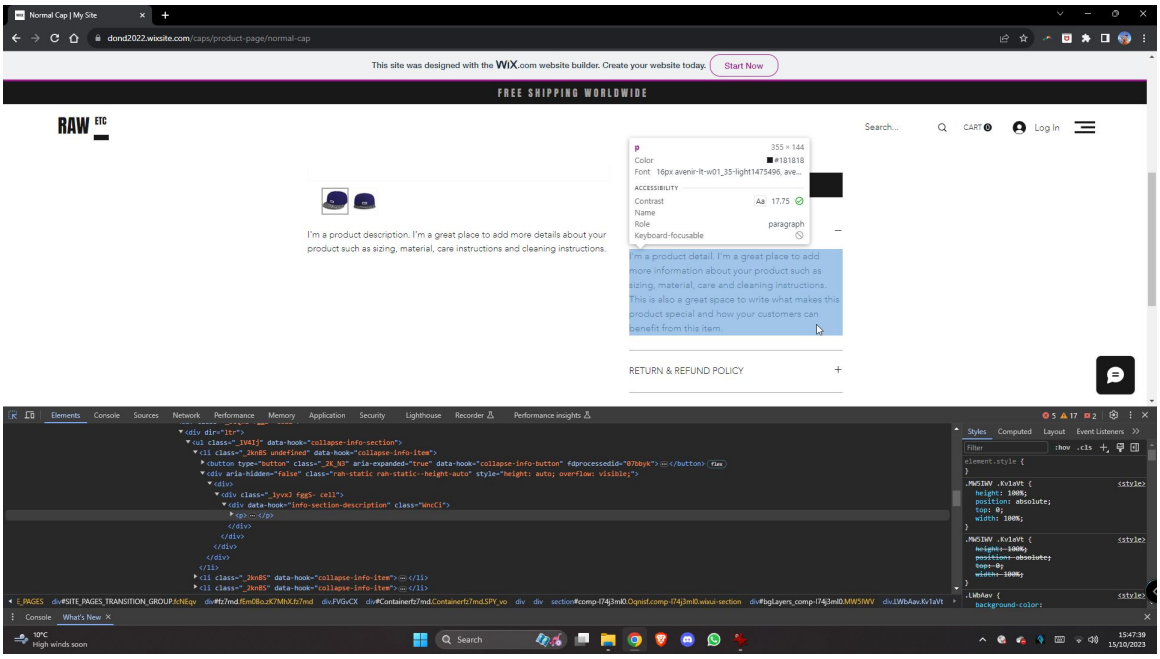

Figure 15: Product description in Google Chrome

The initial tag may be:

```
<p>...</p>
```

Since this tag lacks a class, look to the parent:

```
<div class="WncCi" data-hook="">...</div>
```

Use:

```
"sale_item_description_regex": {
    "name": "div",
    "class_": "WncCi"
}
```

4.5 seller_name_regex

Use the selector tool to find the element containing the seller's name.
This is similar to the profile name regex in Section 3.8.

4.6 seller_url_regex

Use the selector tool to find the hyperlink `<a href="...">` pointing to the seller's page.
Similar to the profile link regex in Section 3.9.

4.7 seller_description_regex

If the seller includes a description or bio, extract its surrounding tag. This follows the same approach as the item description regex (see Section 4.4).

4.8 date_regex

Find the tag that contains the date the item was posted. This is similar to the method in Section 3.10.

4.9 price_regex

This field captures the item's price.

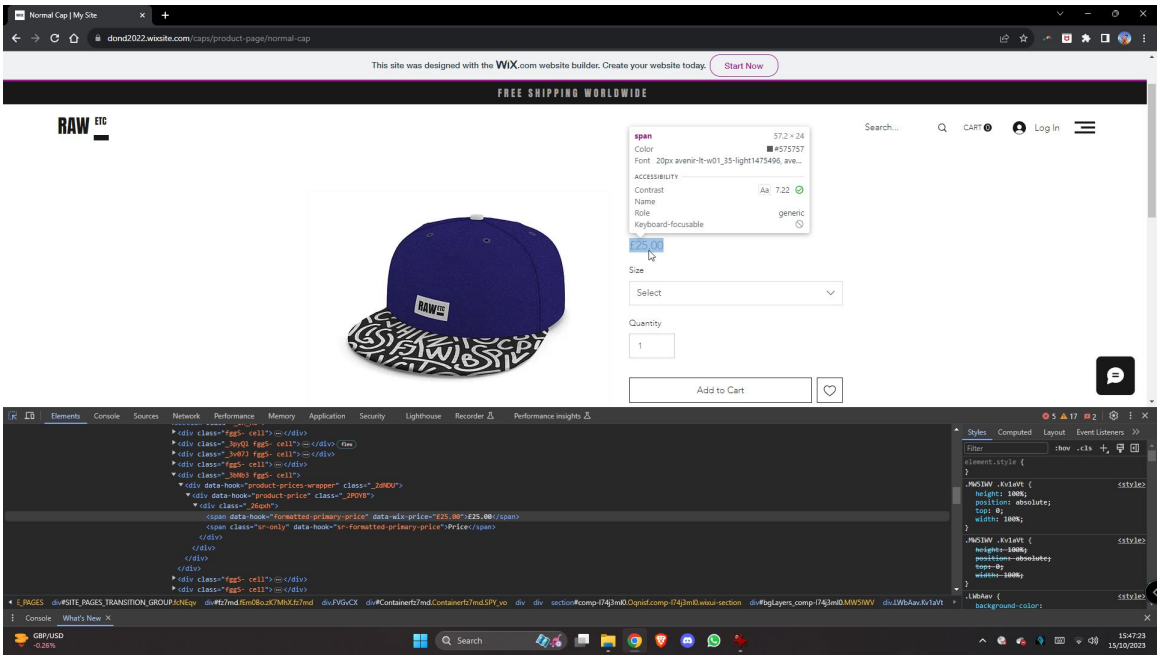

Figure 16: Product price in Google Chrome

In this case, the highlighted HTML is:

```
<span data-hook="formatted-primary-price" data-wix-price="">...</span>
```

Unlike most cases, this website uses `data-hook` instead of `class`. We use this tag to isolate the numerical price only.
Use:

```
"price_regex": {
    "name": "span",
    "data-hook": "formatted-primary-price"
}
```

4.10 review_block_regex

Identifies the HTML block containing a full user review. Use the selector tool to highlight the entire review card.
This is similar to `block_regex` in Section 3.5.

4.11 review_username_regex

Find the tag that contains the reviewer's username.

4.12 review_date_regex

Find the tag that shows when the review was posted.

4.13 review_title_regex

Captures the review title, if available. This field is optional.

4.14 review_description_regex

Captures the main text of the review. Select the block that contains the full review content.

4.15 review_link_regex

If the reviewer's username links to a profile, capture the `<a href="...">` tag.

4.16 attributes_regex

This optional section is used to extract extra details specific to the listing, such as rating, tags, badges, etc.
Example:

```
"attributes_regex": {
    "rating": {
        "name": "div",
        "class_": "rating-stars"
    },
    "tag": {
        "name": "span",
        "class_": "item-tag"
    }
}
```