The main processes in the development of load-testing scripts

Contents

Configuring JMeter

Test Plan is a parent element in JMeter under which all other elements are attached. It is a default element.

When opening JMeter, we get an empty Test Plan. The first thing to do is to add 'Thread Group' element. This will be the main workflow where scripts will be recorded and additional JMeter elements will be added.

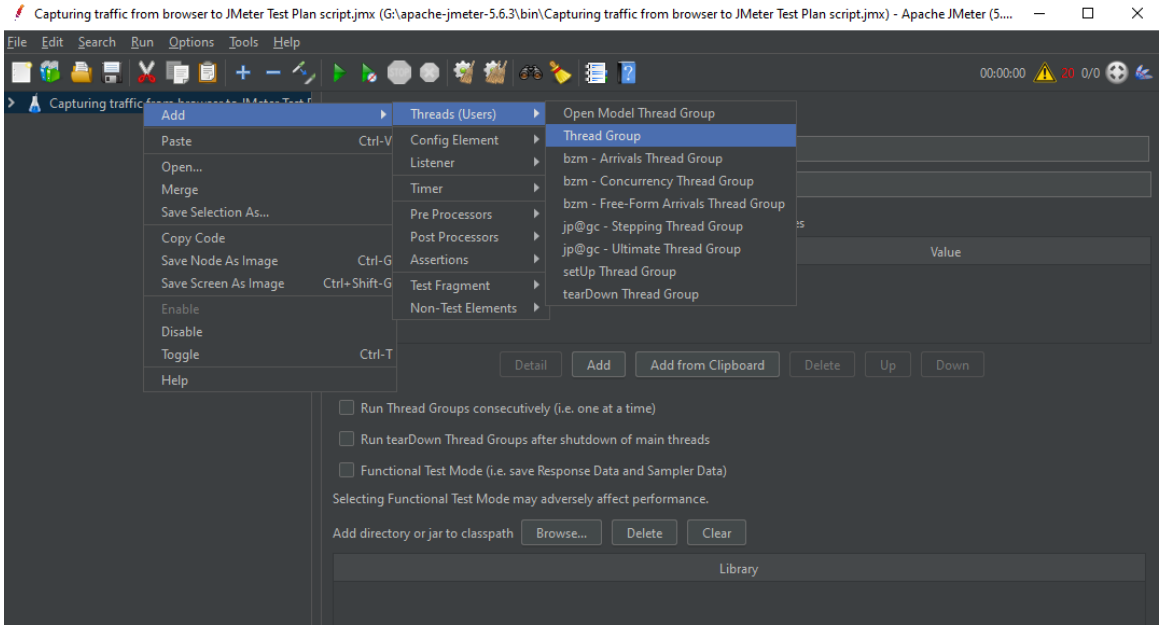


Figure 1 Thread Group

After 'Thread Group' has been created, add following elements from 'Config Element' section:

- HTTP Cookie Manager
- HTTP Cache Manager
- HTTP Request Defaults
- User Defined Variables

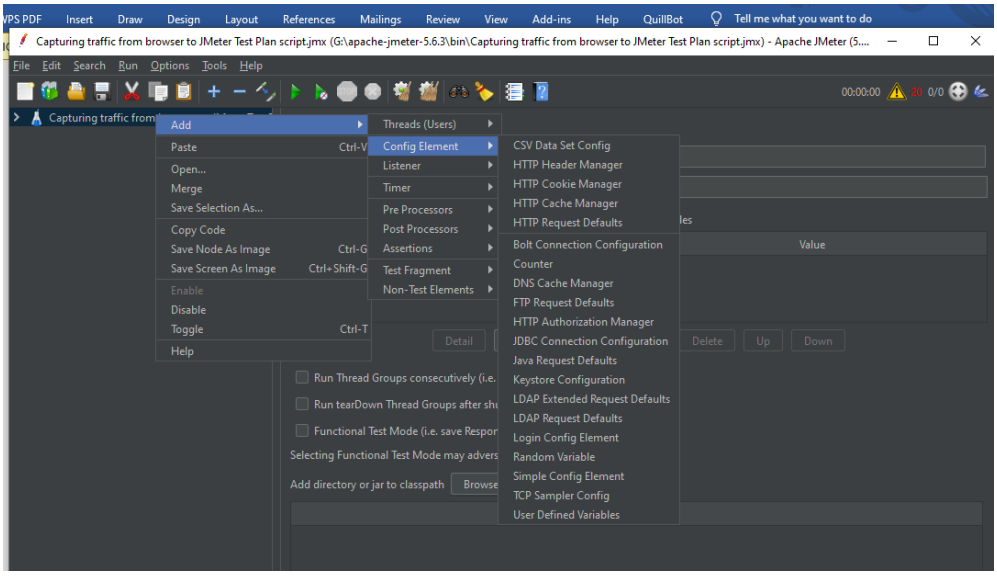


Figure 2 Config Element

1. **HTTP Cookie Manager:**
   - **Purpose:** Manages cookies sent by the server and adds them to subsequent HTTP requests.
   - **Path in JMeter:** Add -> Config Element -> HTTP Cookie Manager

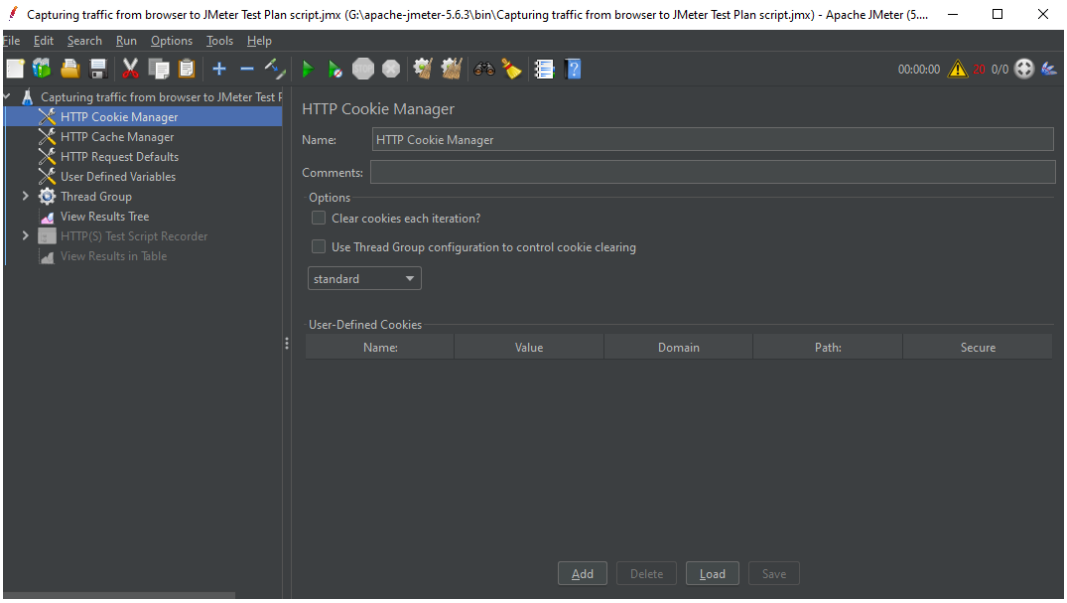


Figure 3 HTTP Cookie Manager

2. **HTTP Cache Manager:**
   - **Purpose:** Simulates browser cache behavior by caching server responses, reducing the need to re-download resources.
   - **Path in JMeter:** Add -> Config Element -> HTTP Cache Manager

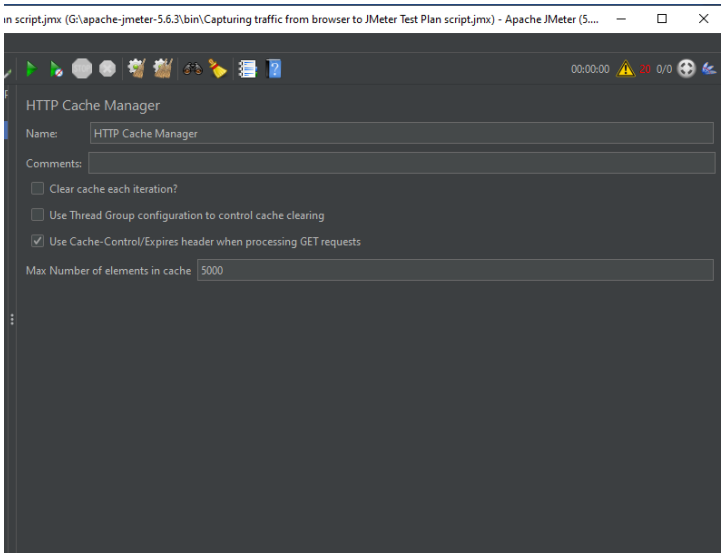


Figure 4 HTTP Cache Manager

3. **HTTP Request Defaults:**
   - **Purpose:** Sets default values for HTTP requests, such as server IP, port, protocol, and path.
   - **Path in JMeter:** Add -> Config Element -> HTTP Request Defaults

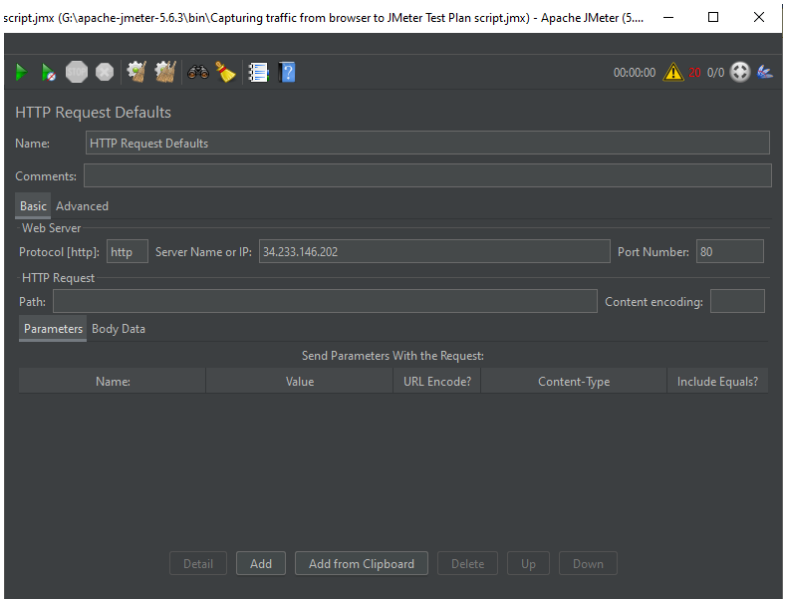
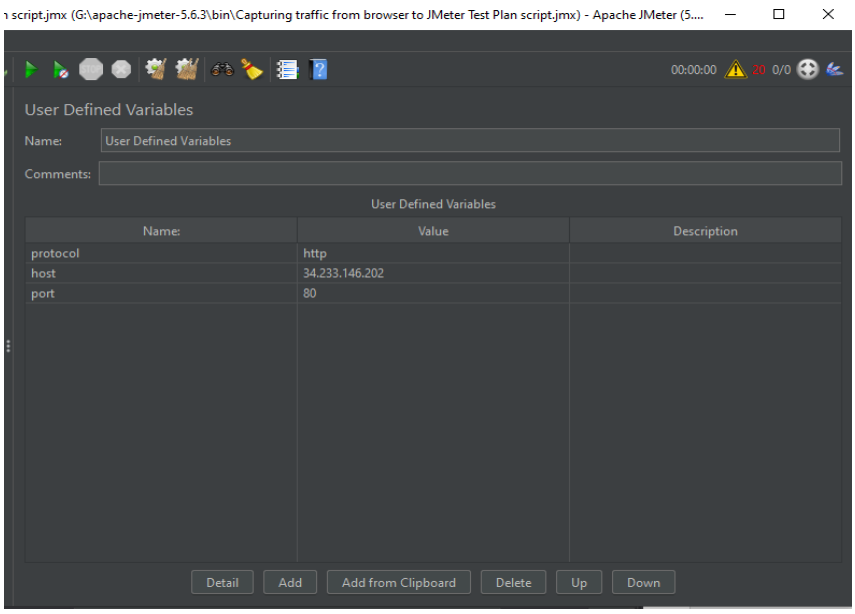


Figure 5 HTTP Request Defaults

4. **User Defined Variables:**
   - **Purpose:** Defines variables with static values that can be reused across multiple elements within the test plan.
   - **Path in JMeter:** Add -> Config Element -> User Defined Variables



HTTP Request:

- **Purpose:** Sends an HTTP/HTTPS request to a web server and receives a response. This is the core element used to simulate user interaction with a web application or API.
- **Path in JMeter:** To add an HTTP Request in JMeter:
    - **Step:** Add -> Sampler -> HTTP Request

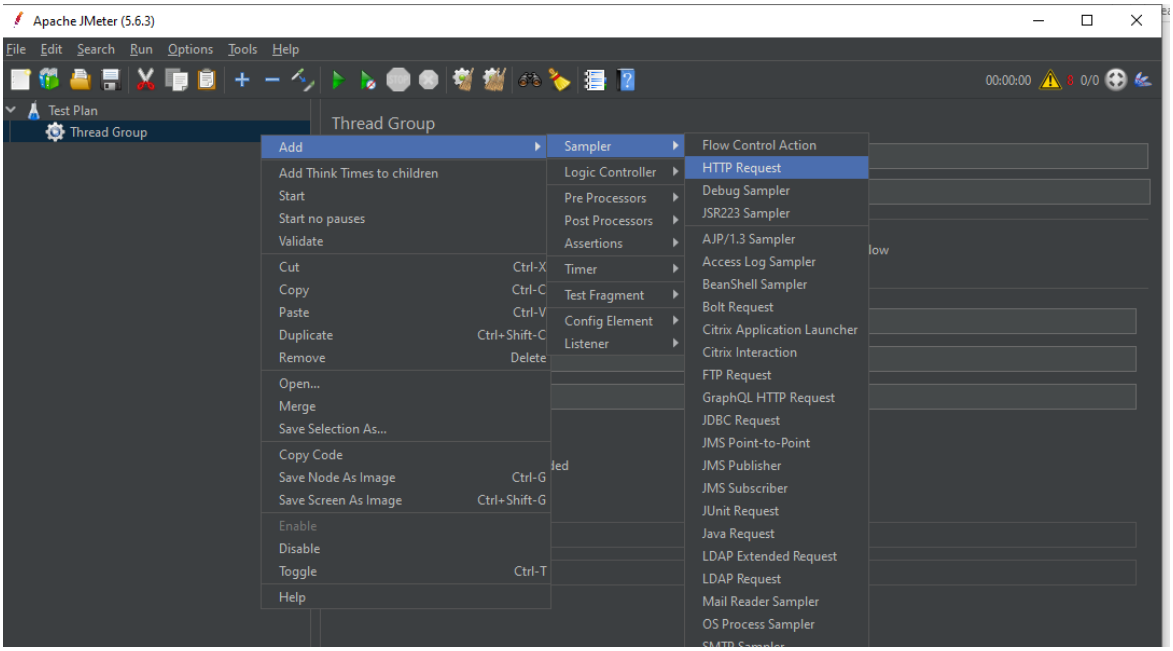


Figure 6 HTTP Request

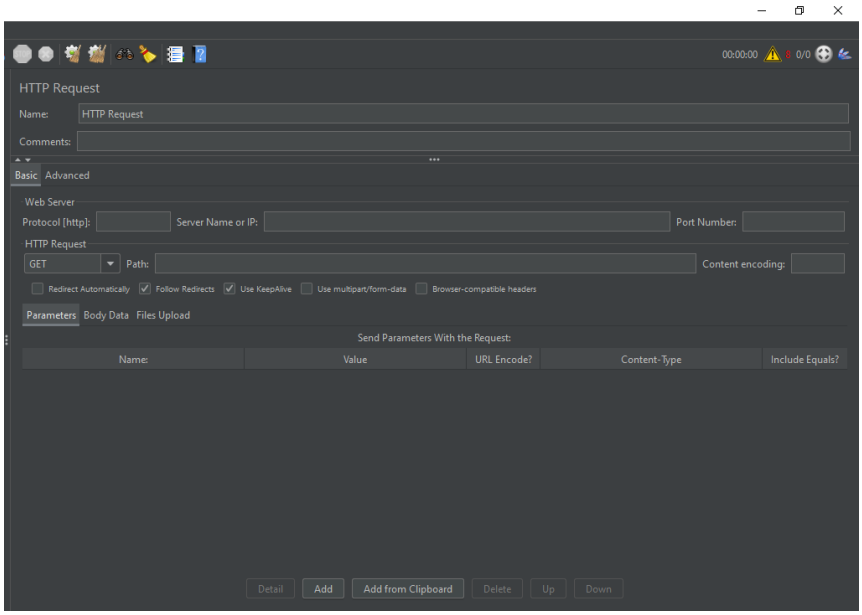


Figure 7 HTTP Request Configuration

Transaction Controller:

Purpose: It measures the overall response time of all samplers within the transaction, providing a consolidated view of performance metrics for a specific business transaction or user action.

Path in JMeter: To add a Transaction Controller:

Step: Add -> Logic Controller -> Transaction Controller

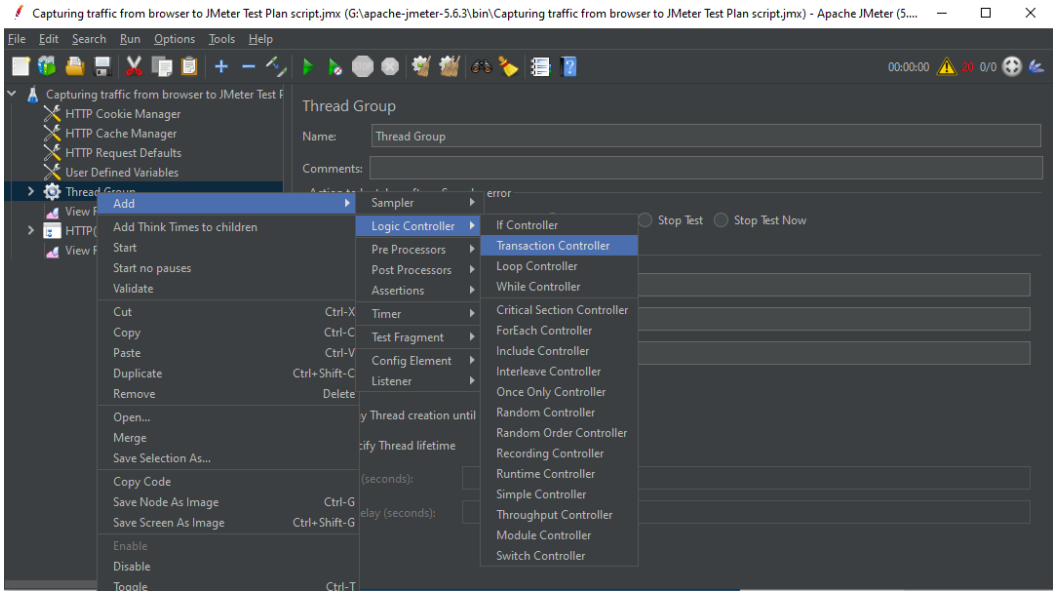


Figure 8 Adding Transaction Controller

HTTP(S) Test Script Recorder:

- **Purpose:** The HTTP(S) Test Script Recorder in JMeter is used to record HTTP or HTTPS requests sent by a browser or any other application. It generates JMeter scripts based on the recorded interactions, allowing testers to replicate user actions and scenarios in their performance tests.
- **Path in JMeter:** To configure and use the HTTP(S) Test Script Recorder:
    - **Step:** Add -> Non-Test Elements -> HTTP(S) Test Script Recorder

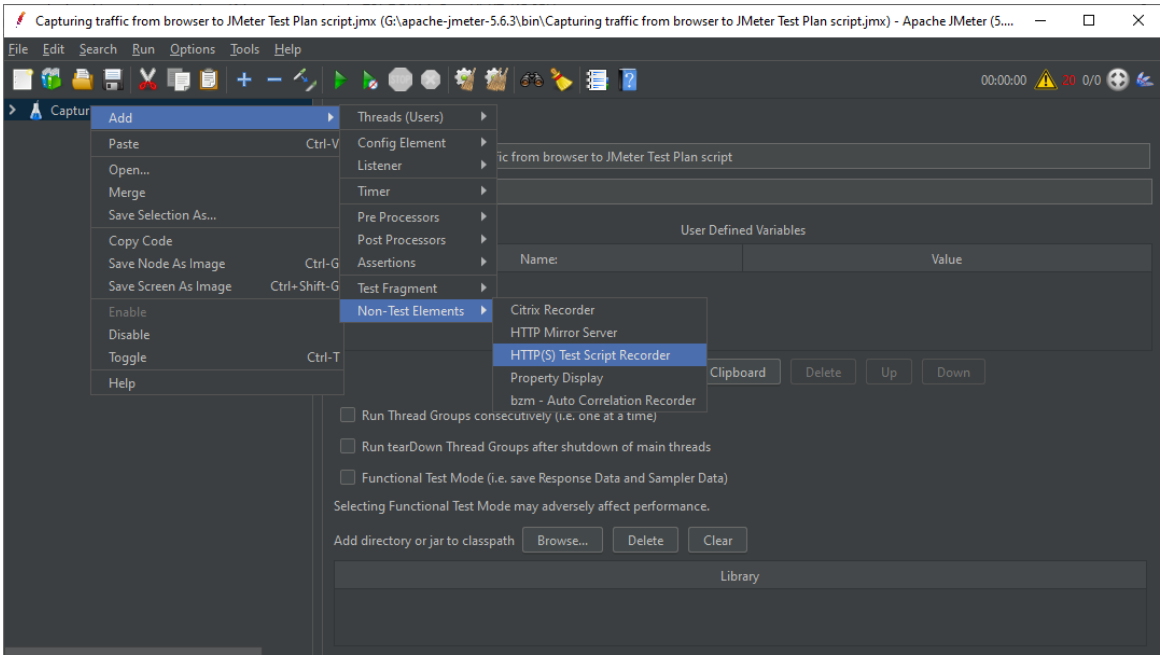


Figure 9 Adding HTTP(S) Test Script Recorder

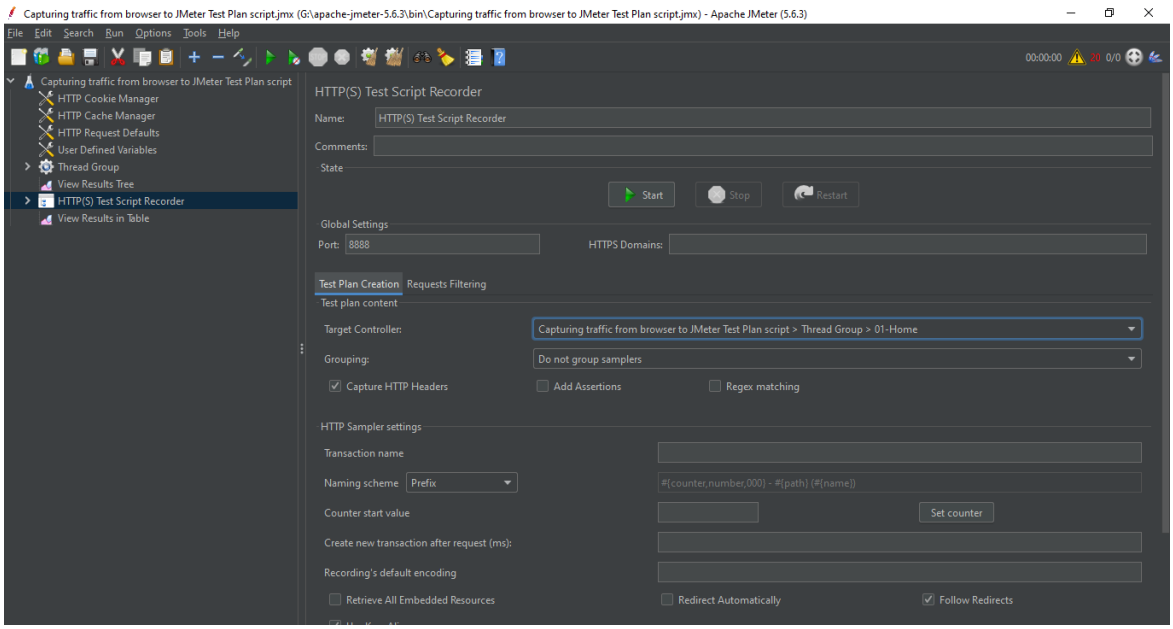


Figure 10 Recording traffic by HTTP(S) Test Script Recorder

JMeter Root CA Certificate:

- **Purpose:** The JMeter Root CA Certificate is a self-signed certificate generated by JMeter when recording HTTPS traffic using the HTTP(S) Test Script Recorder. It allows JMeter to intercept and decrypt HTTPS traffic for recording purposes.
- **Path in JMeter:** The JMeter Root CA Certificate is generated and managed automatically when configuring the HTTP(S) Test Script Recorder. Here's how to manage it:
    - **Step 1:** Configure the HTTP(S) Test Script Recorder in JMeter (Add -> Non-Test Elements -> HTTP(S) Test Script Recorder).
    - **Step 2:** Specify the port number for the proxy server (default is 8888).
    - **Step 3:** Start the HTTP(S) Test Script Recorder. JMeter will prompt to install the JMeter Root CA Certificate.

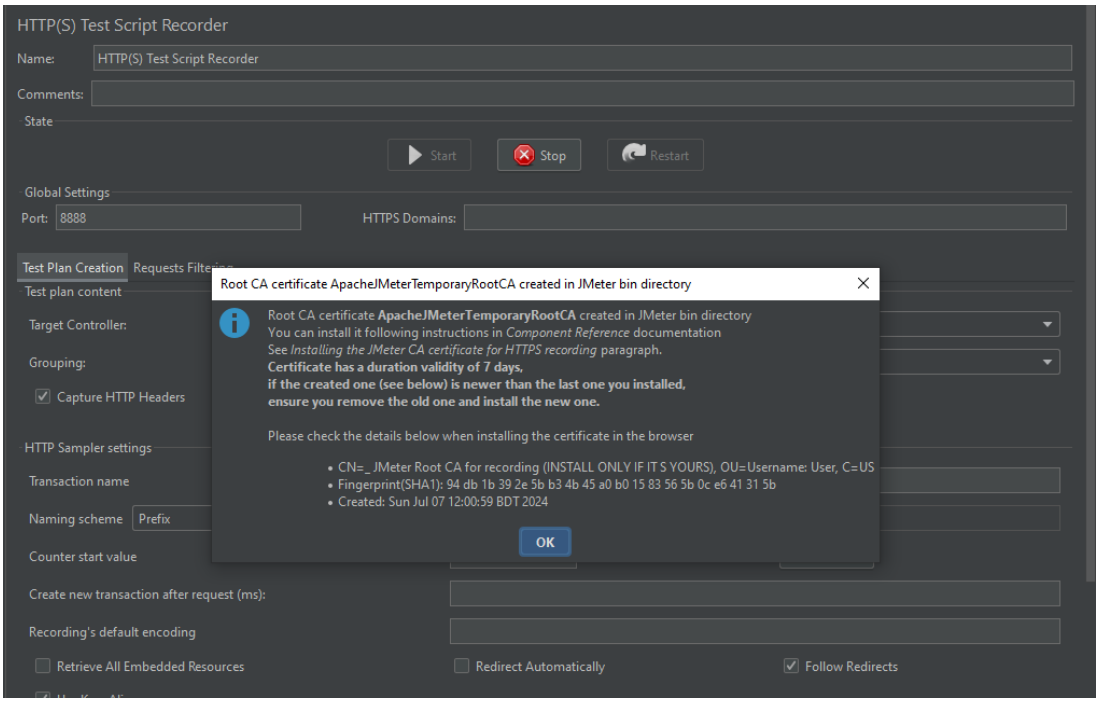


Figure 11 JMeter Root CA Certificate

Listeners - View Result Tree:

- **Purpose:** The View Result Tree listener in JMeter is used to view and analyze individual HTTP request and response details during a test execution. It provides real-time insights into the execution of samplers and helps in debugging and optimizing JMeter scripts.
- **Path in JMeter:** To add the View Result Tree listener:
    - **Step :** Add -> Listener -> View Results Tree

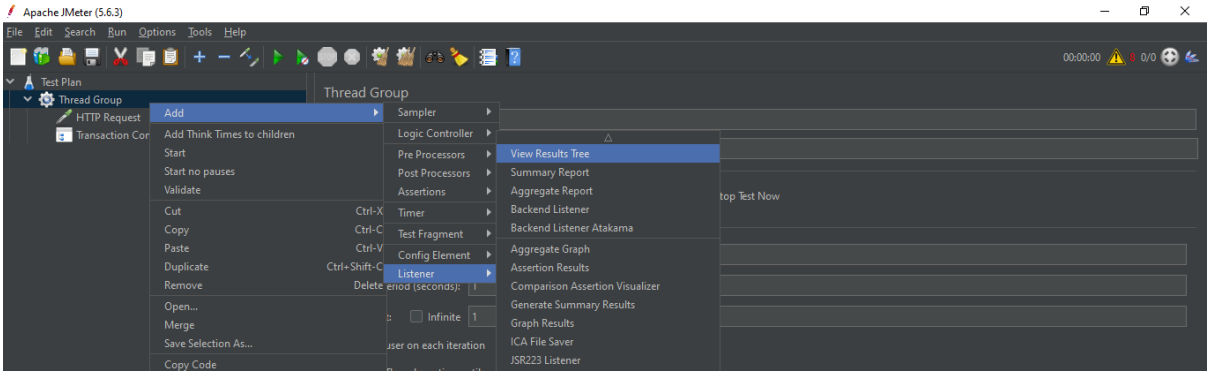


Figure 12 Adding View Results Tree

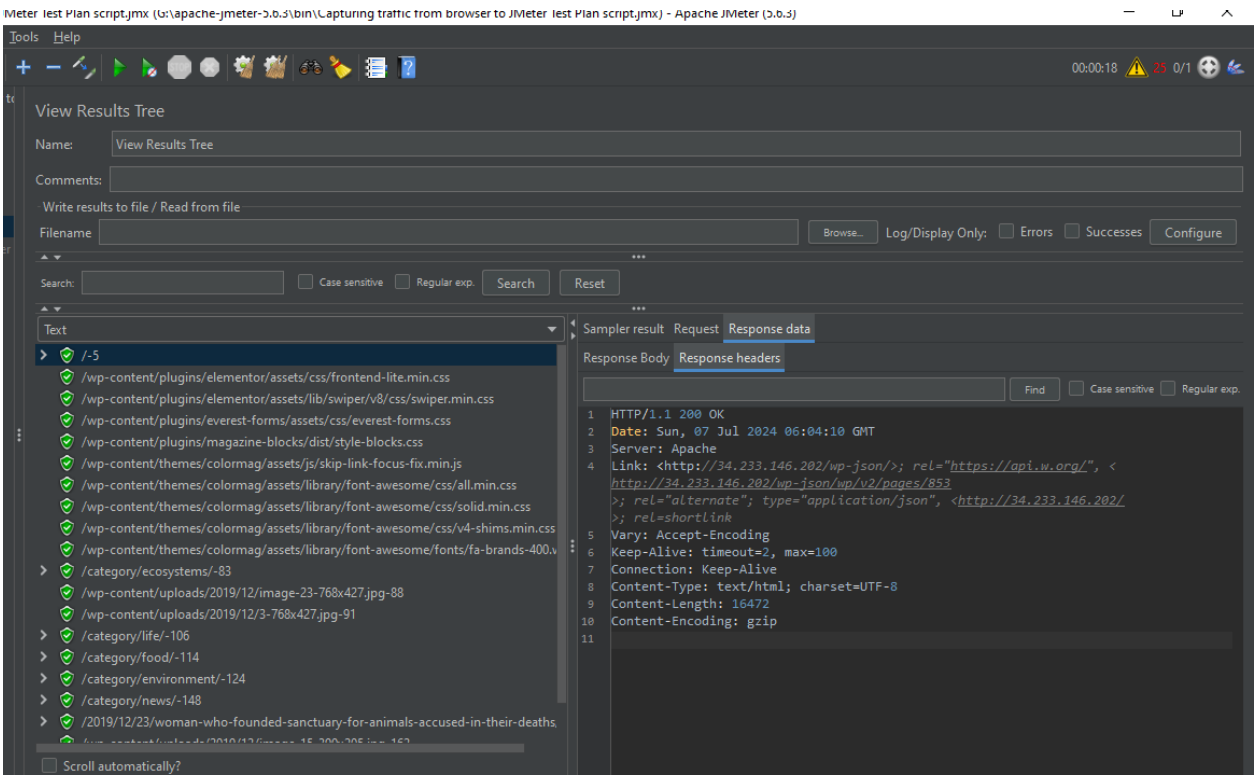


Figure 13 Running View Results Tree

Configuring Fiddler

Import Certificate:

1. **Open Options:** I went to "Tools -> Options" in Fiddler.
2. **Navigate to HTTPS Tab:** I selected the "HTTPS" tab.
3. **Enable HTTPS Decryption:** I checked the box next to "Decrypt HTTPS traffic".
4. **Ignore Certificate Errors:** I also checked the box next to "Ignore server certificate errors (unsafe)".
5. **Export Root Certificate:** To use Fiddler as a proxy across devices, I exported the Root Certificate to my desktop.

Set Proxy:

1. **Configure Proxy Port:** In the "Connections" tab of Fiddler's Options, I set the proxy port to "8888" (different from JMeter's port).
2. **Restart Fiddler:** After making these changes, I restarted Fiddler for the settings to take effect.

These steps allowed me to set up Fiddler as a proxy, enabling me to capture and analyze HTTP and HTTPS traffic from my browser effectively.

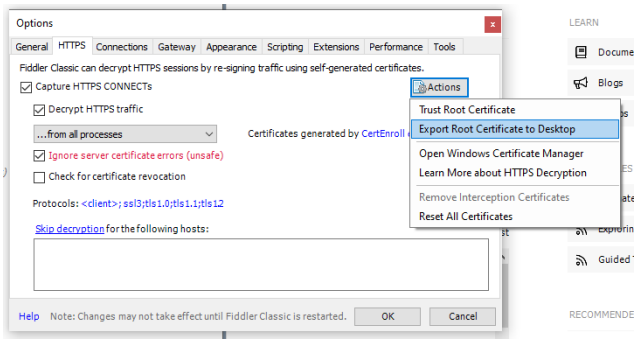


Figure 14 Fiddler Configuration

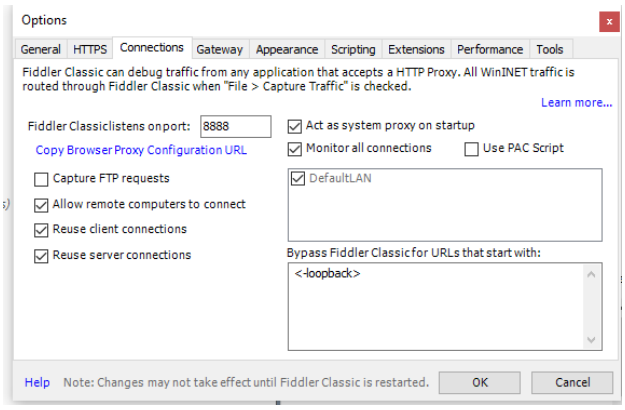


Figure 15 Set up Connections

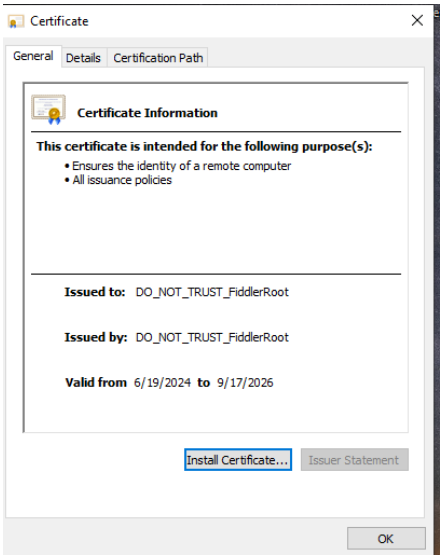


Figure 16 Certificate Exported from Fiddler

Configuring Firefox

Proxy Configuration:

1. **Open Firefox Settings:** I navigated to the Settings menu in Firefox.
2. **Navigate to Network Settings:** I scrolled down to "Network Settings".
3. **Manual Proxy Configuration:** I selected "Manual proxy configuration".
4. **Set HTTP Proxy:** I set the HTTP Proxy to "localhost".
5. **Set Port:** I configured the port to "8888" (for JMeter) or "8887" (for Fiddler) and checked "Also use this proxy for HTTPS".

Allow Hijacking for localhost:

1. **Access about:config:** I entered "about

   " in the browser's address bar.

2. **Change Configuration:** I found the "network.proxy.allow_hijacking_localhost" parameter and set its value to "true".

Import Certificates:

1. **Certificate Manager:** In Firefox Settings, I searched for the "Certificates" section.
2. **Import JMeter Certificate:** I imported the JMeter certificate generated in the HTTP(S) Test Script Recorder settings.
3. **Import Fiddler Certificate:** Similarly, I imported the Fiddler certificate created in Fiddler's HTTPS options.

After completing these steps, Firefox was configured to proxy and record traffic between the browser, JMeter, and Fiddler, enabling effective analysis and testing of web applications.

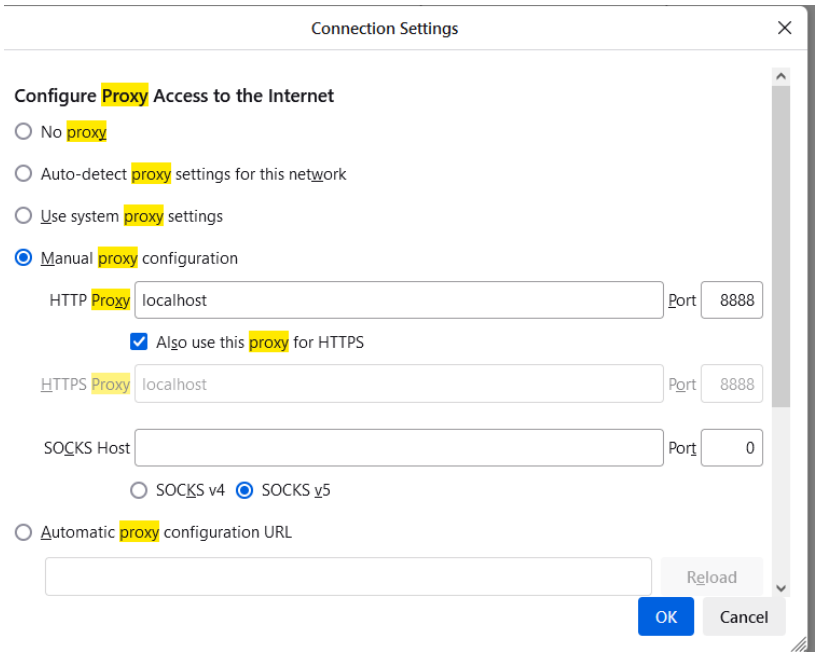


Figure 17 Firefox proxy Setup

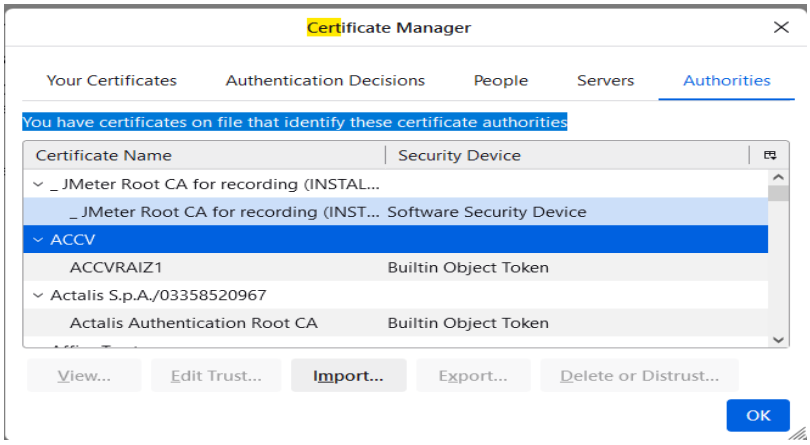


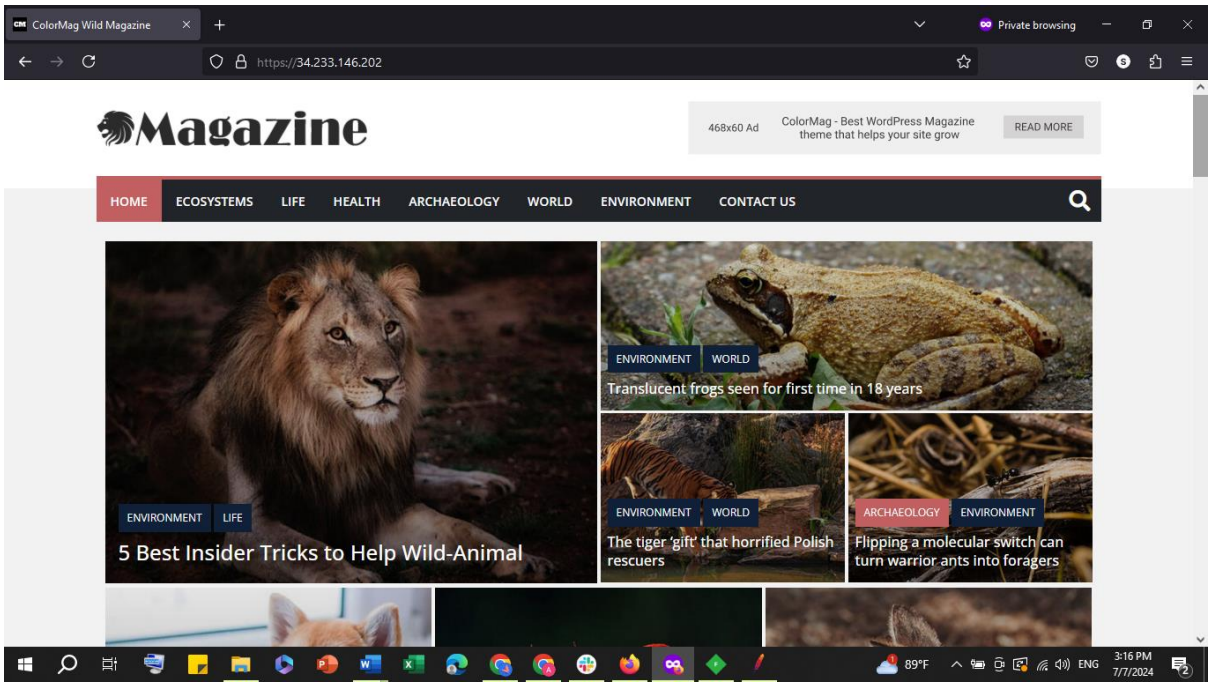
Figure 18 Importing JMeter and Fiddler certificate

Capturing traffic from the browser to Fiddler

To proxy traffic from the browser to Fiddler, it's necessary to set the same port in the browser as in Fiddler. The following steps outline how to proxy traffic from the browser to Fiddler:

General Recommendations

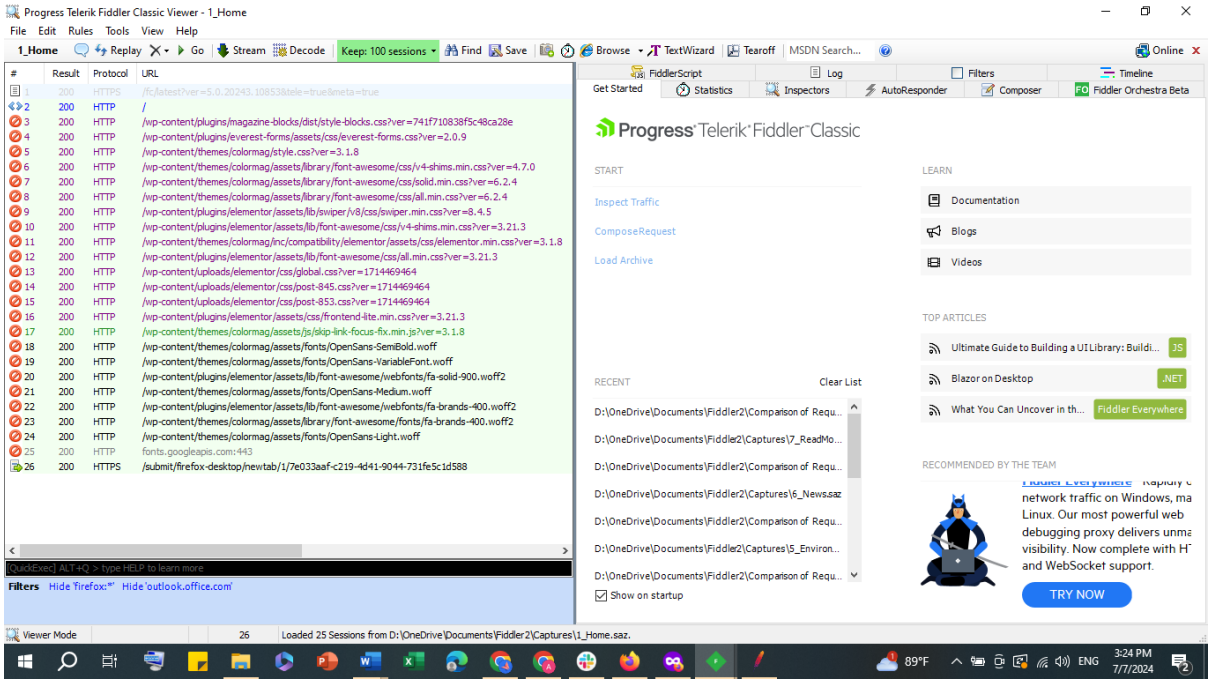
- Use an Incognito Window when proxying your browser traffic.



Steps to Proxy Traffic

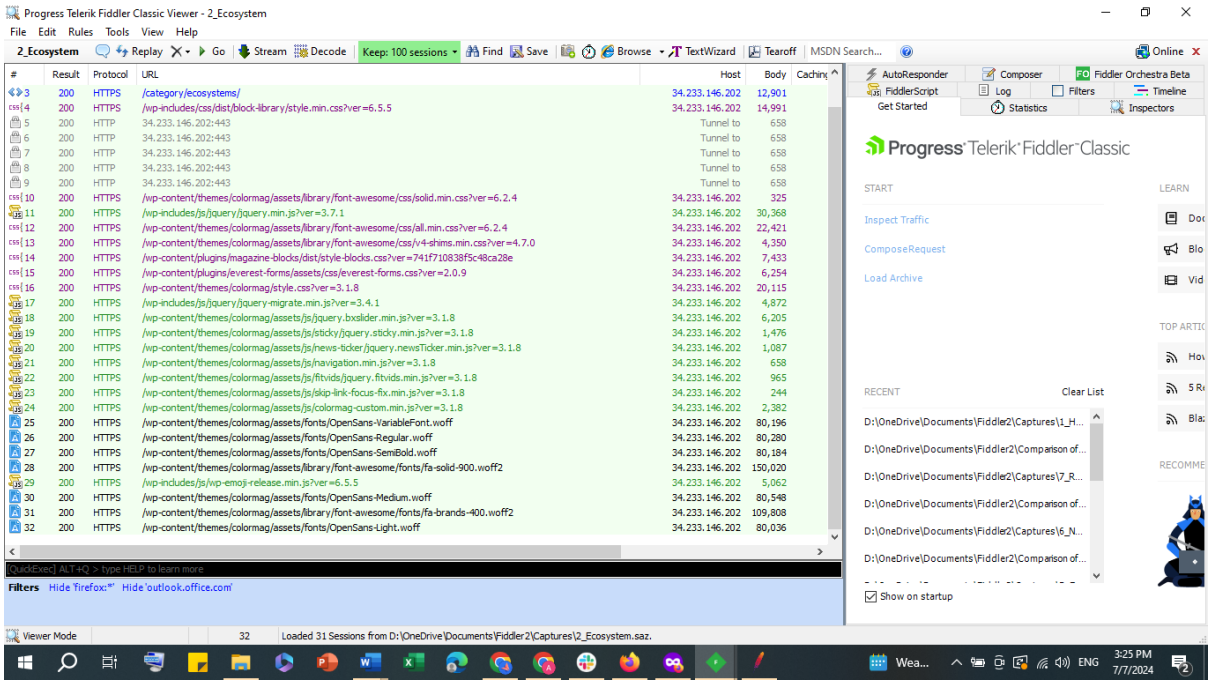
➢ **First Step: Open Main Page**

1. I opened both my browser and Fiddler, ensuring that both were configured to use the same proxy ports.
2. I navigated to the main page on my browser: `http://34.233.146.202`.
3. I checked Fiddler to ensure I could see requests from the `34.233.146.202` host. I was mindful of the website cache and ensured traffic was recorded correctly by refreshing the page (F5) or checking the number of requests via DevTools in the browser.
4. I saved the recorded traffic in Fiddler to a file by selecting `File -> Save -> All Sessions…` and named this file as the first step.



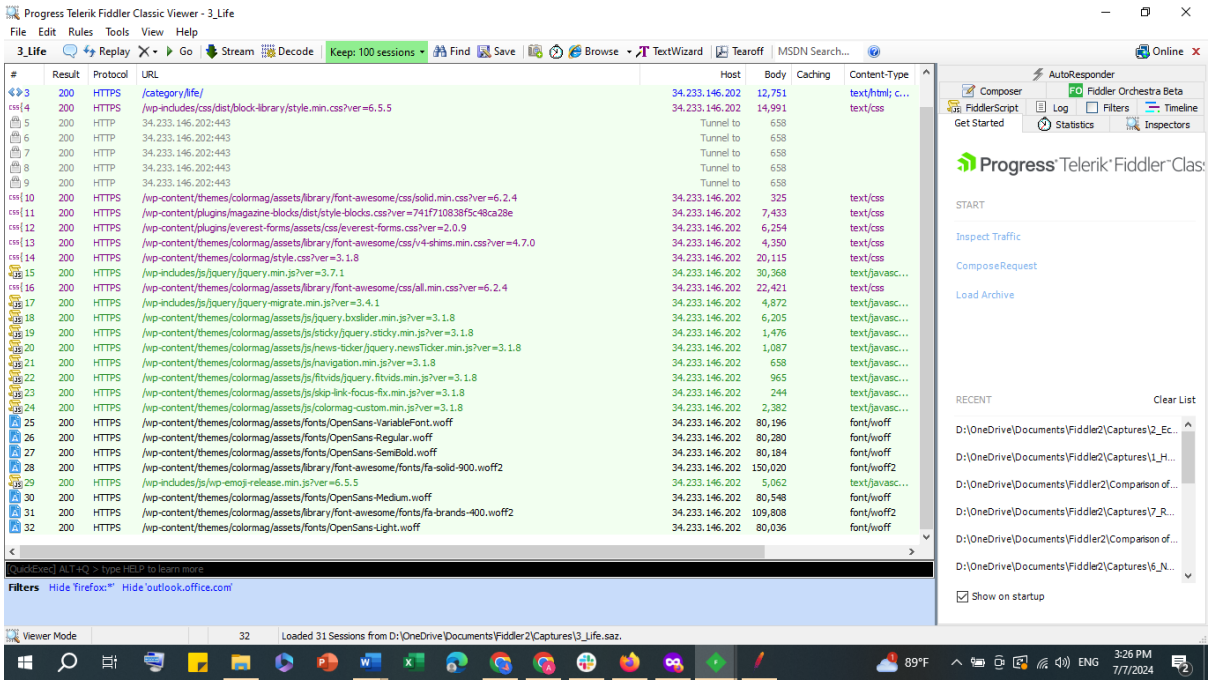
➢ Second Step: Open Ecosystems Page

1. I made sure there were no captured requests in Fiddler by clearing previous sessions.
2. From the open browser on the main page, I navigated to the Ecosystems page.
3. I checked Fiddler to ensure I could see requests from the `34.233.146.202` host. I ensured traffic was recorded correctly by refreshing the page (F5) or checking the number of requests via DevTools in the browser.
4. I saved the recorded traffic in Fiddler to a file by selecting `File -> Save -> All Sessions…` and named this file as the second step.



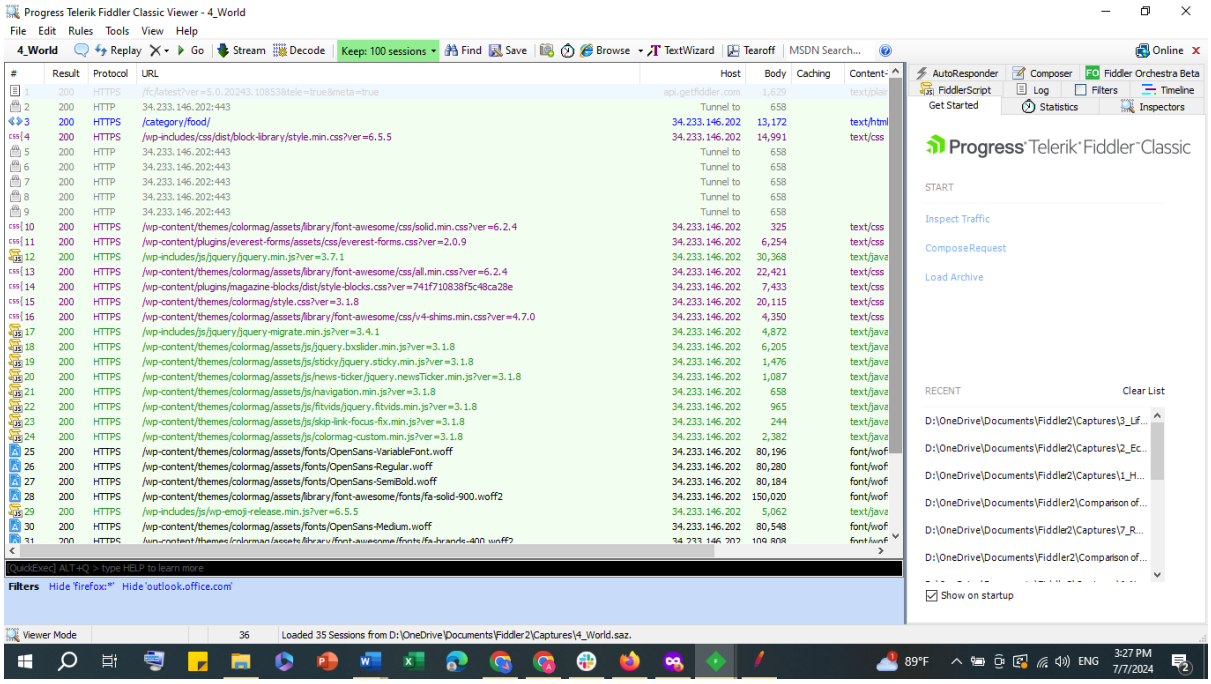
➢ Third Step: Open Life Page

1. I made sure there were no captured requests in Fiddler by clearing previous sessions.
2. From the open browser on the Ecosystems page, I navigated to the Life page.
3. I checked Fiddler to ensure I could see requests from the `34.233.146.202` host. I ensured traffic was recorded correctly by refreshing the page (F5) or checking the number of requests via DevTools in the browser.
4. I saved the recorded traffic in Fiddler to a file by selecting `File -> Save -> All Sessions…` and named this file as the third step.



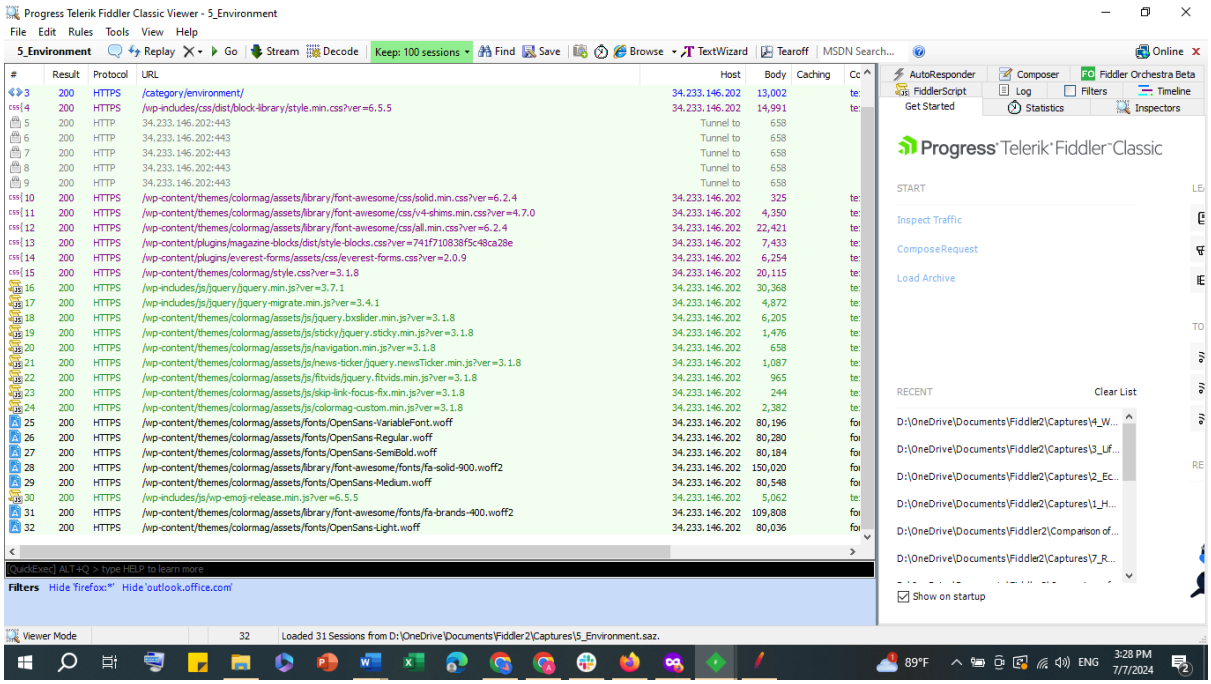
➢ Fourth Step: Open World Page

1. I made sure there were no captured requests in Fiddler by clearing previous sessions.
2. From the open browser on the Life page, I navigated to the World page.
3. I checked Fiddler to ensure I could see requests from the `34.233.146.202` host. I ensured traffic was recorded correctly by refreshing the page (F5) or checking the number of requests via DevTools in the browser.
4. I saved the recorded traffic in Fiddler to a file by selecting `File -> Save -> All Sessions…` and named this file as the fourth step.



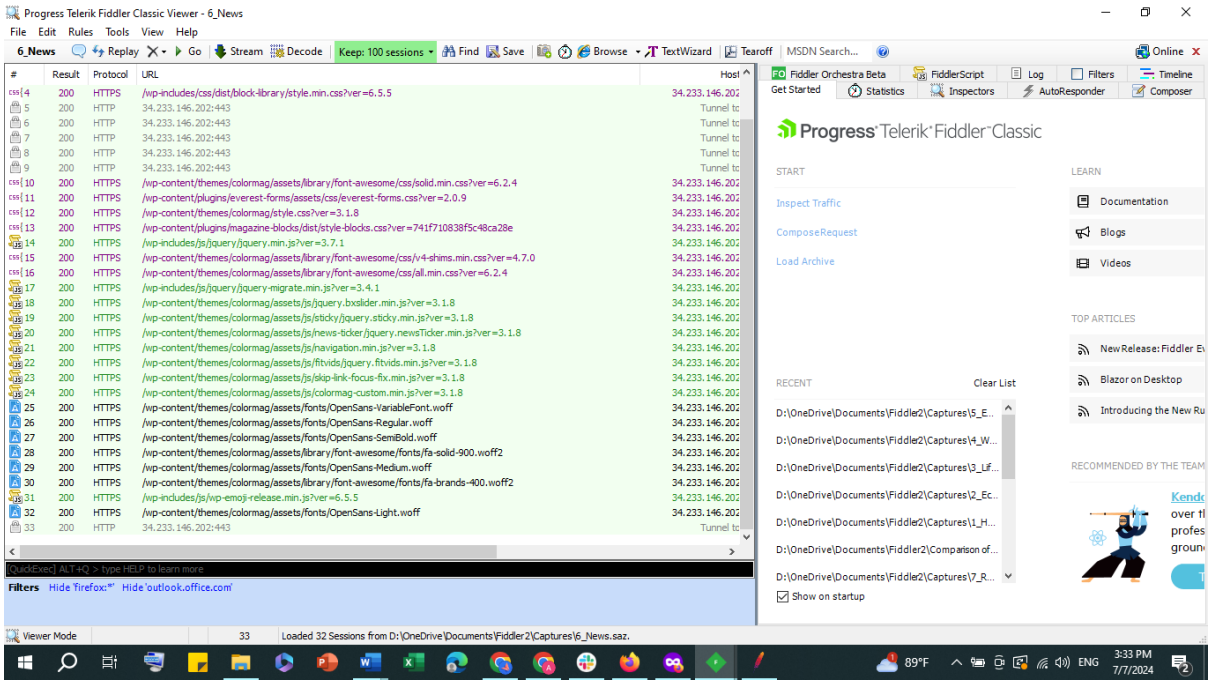
➢ Fifth Step: Open Environment Page

1. I made sure there were no captured requests in Fiddler by clearing previous sessions.
2. From the open browser on the World page, I navigated to the Environment page.
3. I checked Fiddler to ensure I could see requests from the `34.233.146.202` host. I ensured traffic was recorded correctly by refreshing the page (F5) or checking the number of requests via DevTools in the browser.
4. I saved the recorded traffic in Fiddler to a file by selecting `File -> Save -> All Sessions…` and named this file as the fifth step.



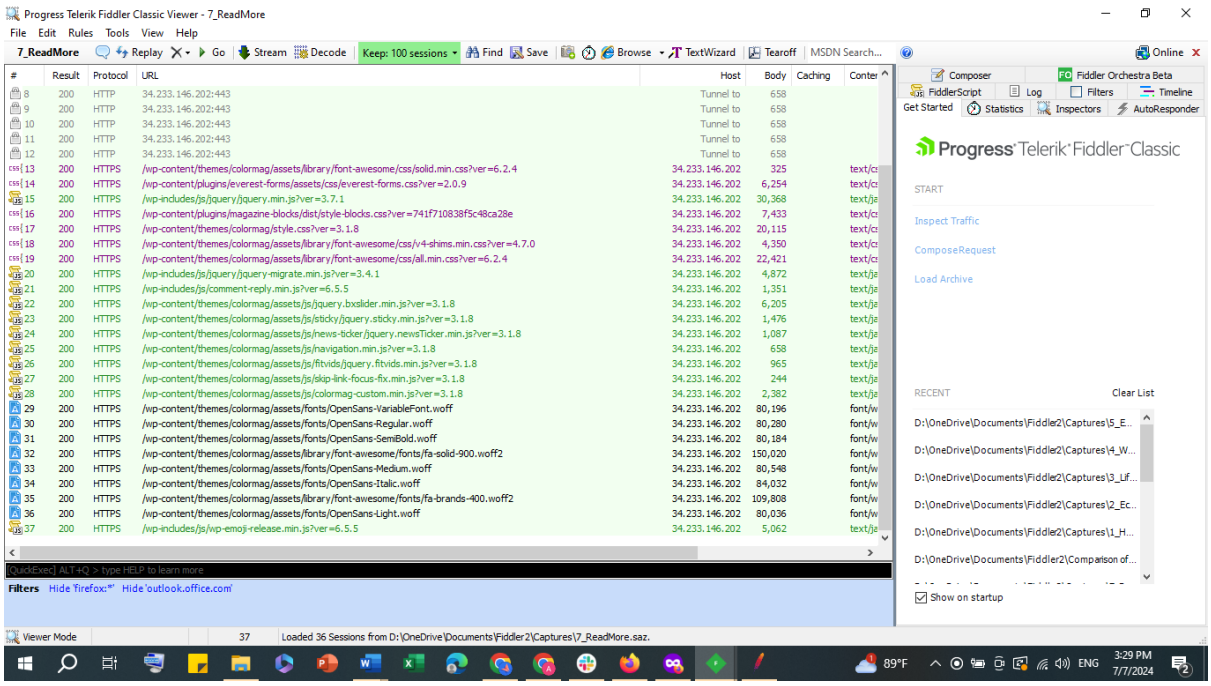
➢ Sixth Step: Open News Page

1. I made sure there were no captured requests in Fiddler by clearing previous sessions.
2. From the open browser on the Environment page, I navigated to the News page.
3. I checked Fiddler to ensure I could see requests from the `34.233.146.202` host. I ensured traffic was recorded correctly by refreshing the page (F5) or checking the number of requests via DevTools in the browser.
4. I saved the recorded traffic in Fiddler to a file by selecting `File -> Save -> All Sessions…` and named this file as the sixth step.



➢ Seventh Step: Open Read More on News Page

1. I made sure there were no captured requests in Fiddler by clearing previous sessions.
2. From the open browser on the News page, I navigated to a random article by clicking "Read more".
3. I checked Fiddler to ensure I could see requests from the `34.233.146.202` host. I ensured traffic was recorded correctly by refreshing the page (F5) or checking the number of requests via DevTools in the browser.
4. I saved the recorded traffic in Fiddler to a file by selecting `File -> Save -> All Sessions…` and named this file as the seventh step.



Additional Notes

- I performed all steps in the browser in the manner that a real user would.
- In addition to the traffic from `34.233.146.202`, there was also third-party request traffic. I removed these third-party requests manually from the file.
- As a result, I obtained seven files with traffic from the browser. These files will be used later to compare the traffic in JMeter.

Capturing traffic from browser to JMeter

To proxy traffic from the browser to JMeter, it's necessary to set the same port in the browser as in JMeter. I will capture the same steps from the browser as in the previous section. Here's how I proceed:

General Recommendations

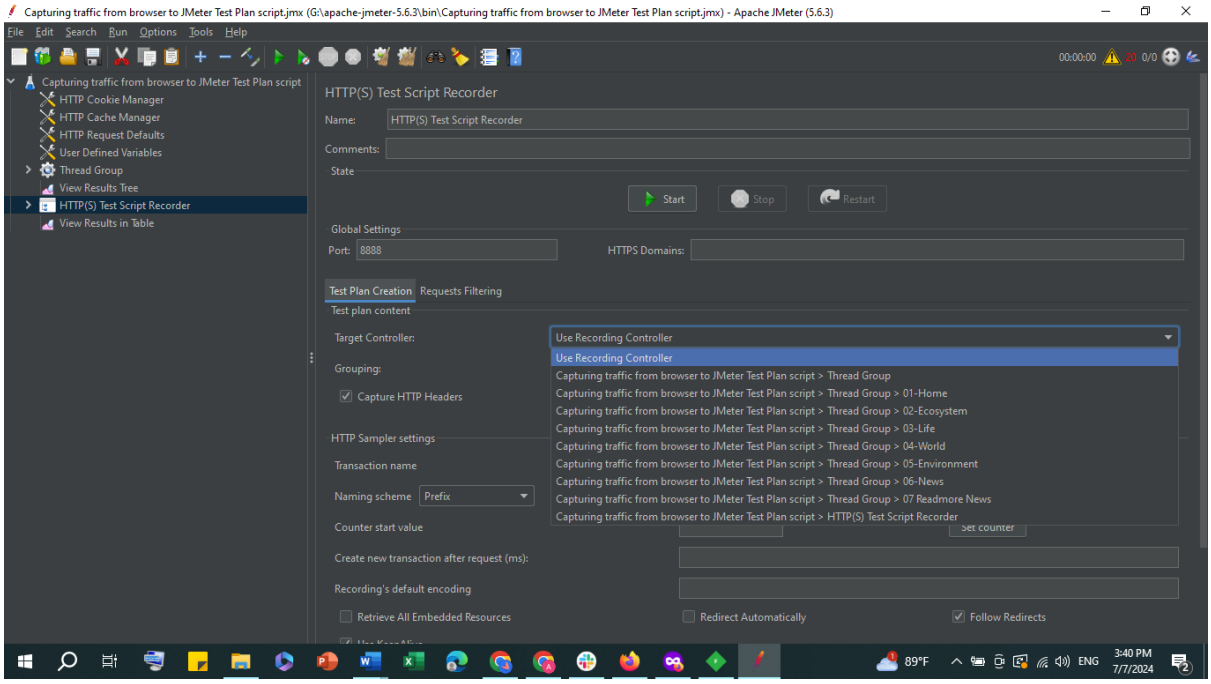
I used an Incognito Window when proxying my browser traffic.

Preconditions on JMeter

1. I prepared the JMX file in advance, where the traffic will be recorded (see section "Configuring JMeter").
2. The number of Transaction Controllers matched the number of steps.

Example of the Test Plan (JMX file) Before Recording Steps

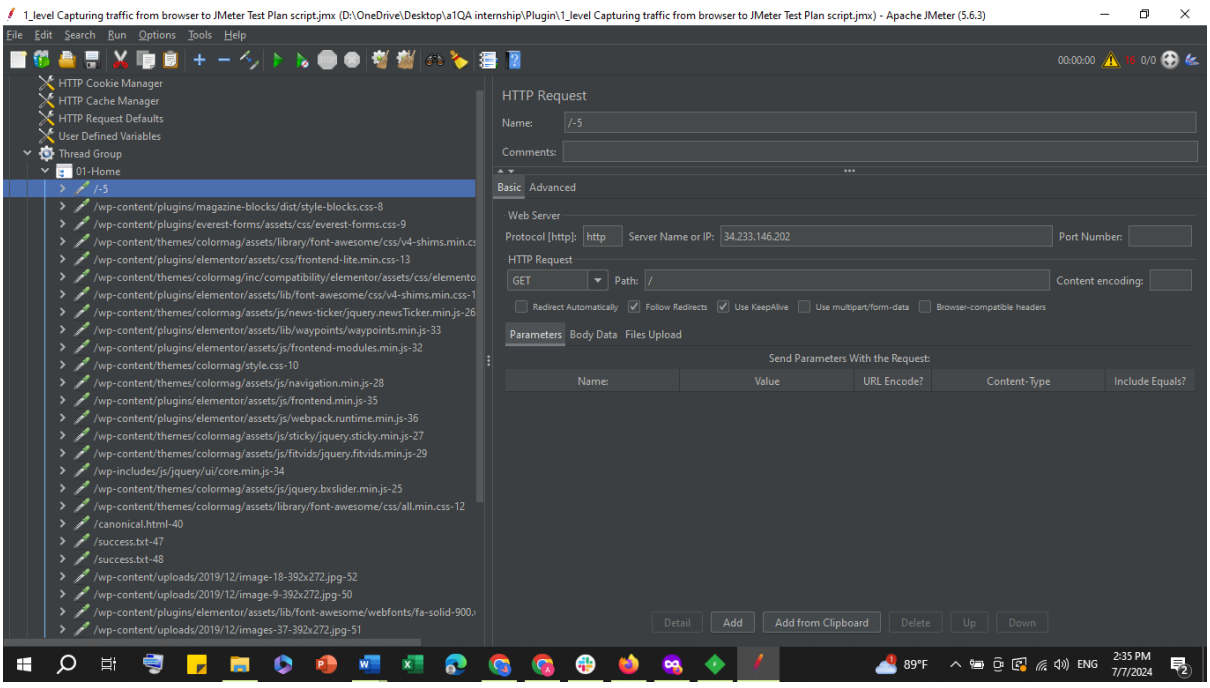
This is how the Test Plan looks before recording steps, ensuring that each step corresponds to a Transaction Controller.



Steps to Capture Traffic

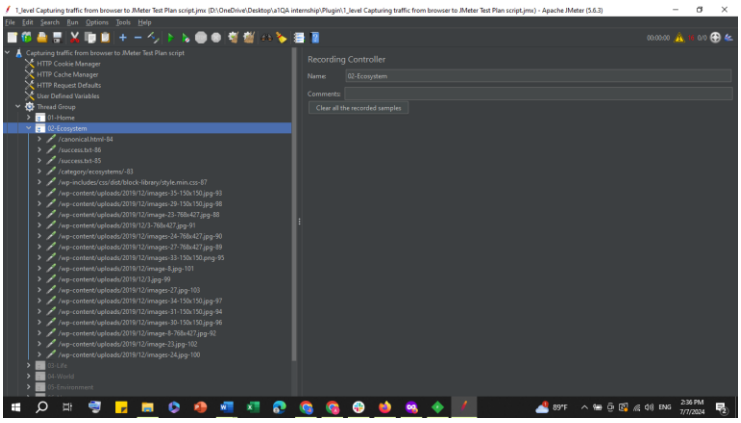
➢ First Step: Open Main Page

1. On the Target Controller, I set the first Transaction Controller.
2. I click on the 'Start' button in the HTTP(S) Test Script Recorder and select 'OK' in the window that appears.
3. I switch to Firefox and navigate to the main page on my browser: `http://34.233.146.202` (using a new Incognito Window).
4. I switch back to JMeter to ensure I see traffic from my host inside the first Transaction Controller.



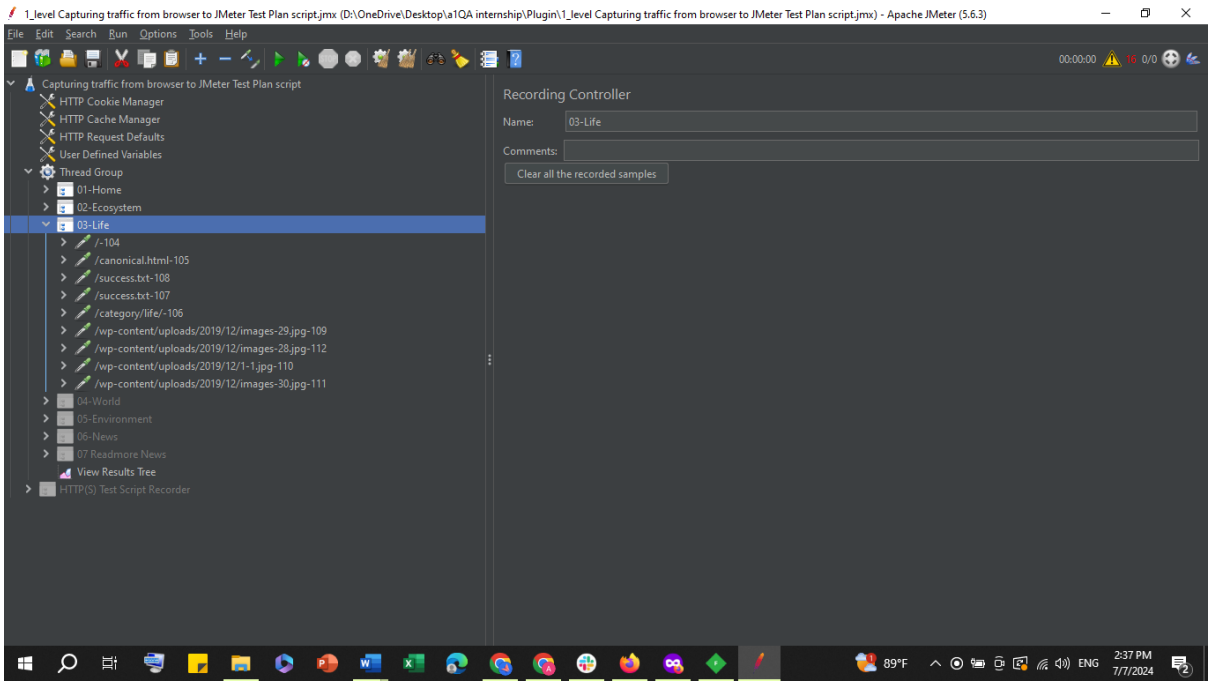
➢ Second Step: Open Ecosystems Page

1. On the Target Controller, I set the second Transaction Controller.
2. I switch to Firefox and navigate to the Ecosystems page from the main page.
3. I switch back to JMeter to ensure I see traffic from my host inside the second Transaction Controller.



➢ Third Step: Open Life Page

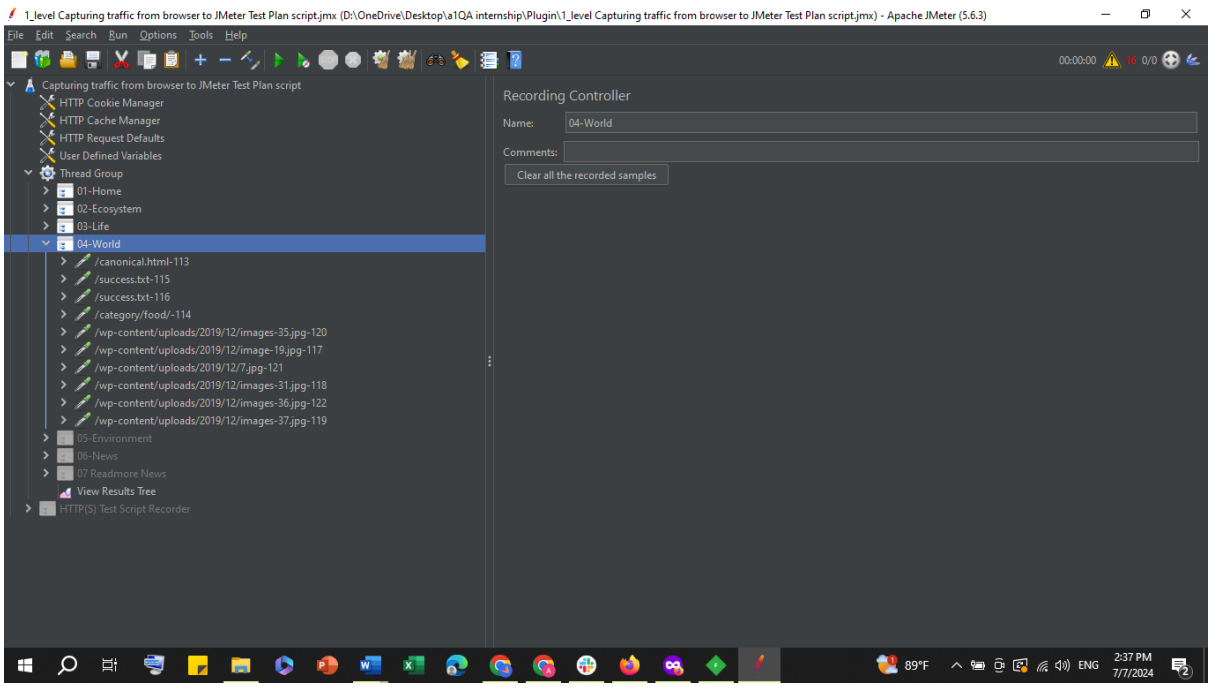
1. On the Target Controller, I set the third Transaction Controller.
2. I switch to Firefox and navigate to the Life page from the Ecosystems page.
3. I switch back to JMeter to ensure I see traffic from my host inside the third Transaction Controller.



➢ Fourth Step: Open World Page

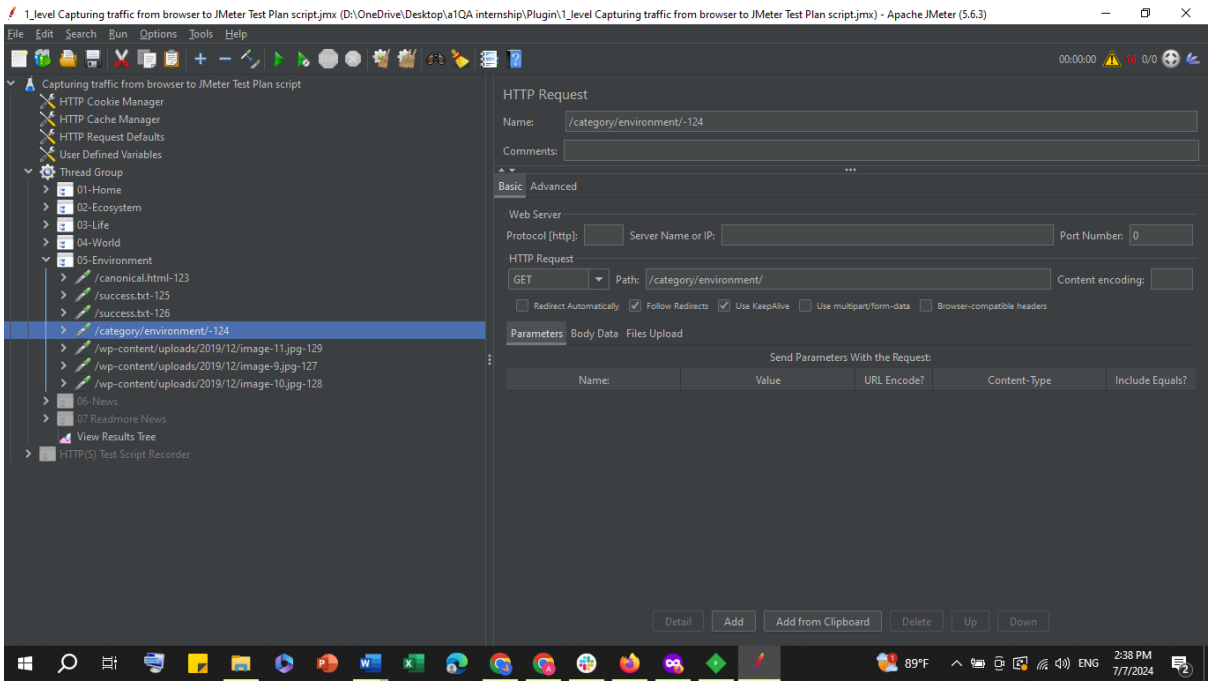
1. On the Target Controller, I set the fourth Transaction Controller.
2. I switch to Firefox and navigate to the World page from the Life page.

3. I switch back to JMeter to ensure I see traffic from my host inside the fourth Transaction Controller.



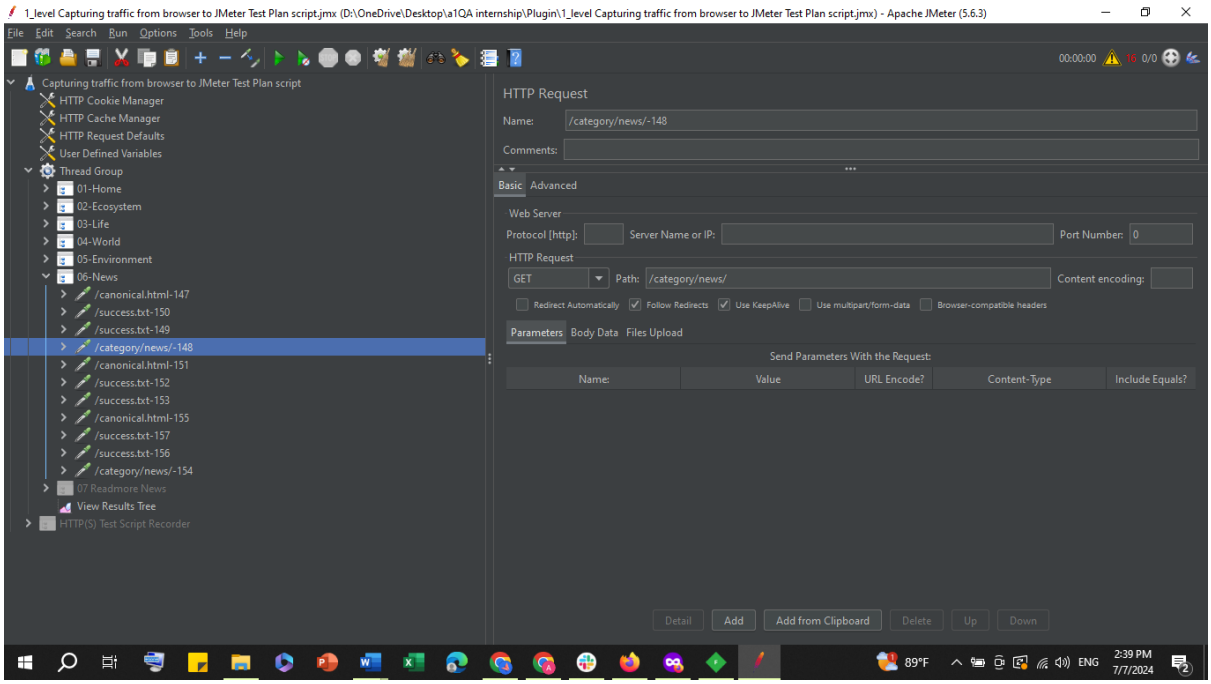
➢ Fifth Step: Open Environment Page

1. On the Target Controller, I set the fifth Transaction Controller.
2. I switch to Firefox and navigate to the Environment page from the World page.
3. I switch back to JMeter to ensure I see traffic from my host inside the fifth Transaction Controller.



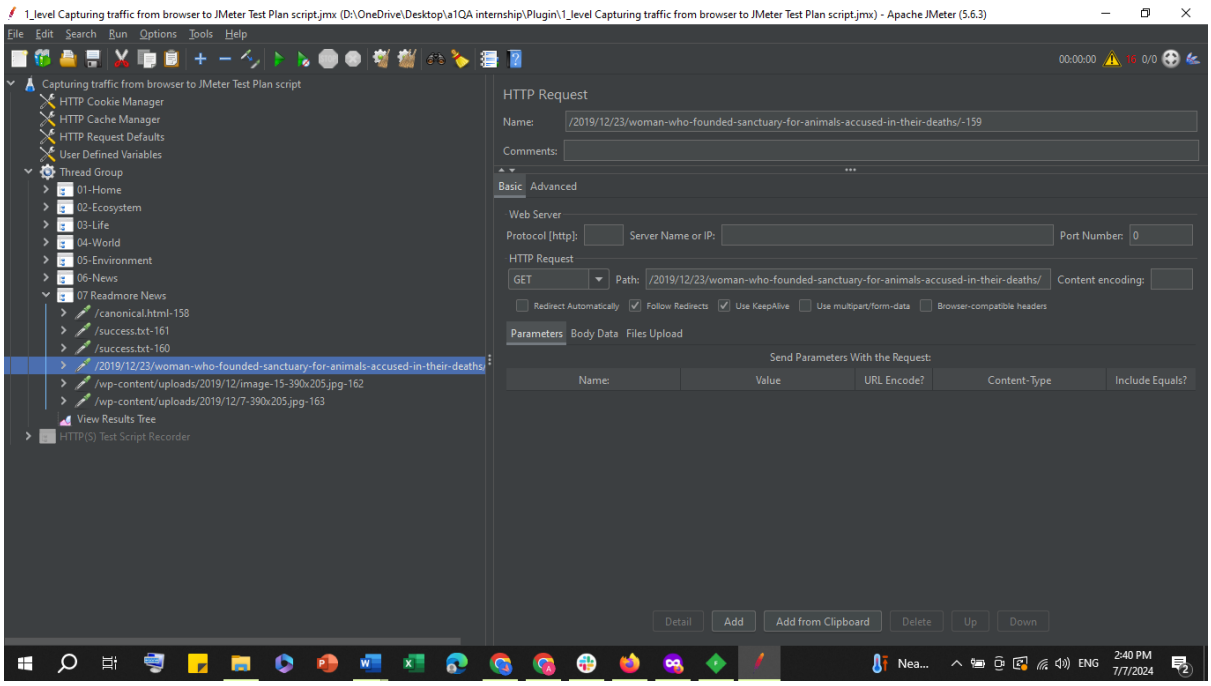
➢ Sixth Step: Open News Page

1. On the Target Controller, I set the sixth Transaction Controller.
2. I switch to Firefox and navigate to the News page from the Environment page.
3. I switch back to JMeter to ensure I see traffic from my host inside the sixth Transaction Controller.



➢ Seventh Step: Open Read More on News Page

1. On the Target Controller, I set the seventh Transaction Controller.
2. I switch to Firefox and navigate to a random article by clicking "Read more" on the News page.
3. I switch back to JMeter to ensure I see traffic from my host inside the seventh Transaction Controller.
4. I click on the ‘Stop’ button in the HTTP(S) Test Script Recorder to stop capturing traffic to JMeter.



Additional Notes

- When loading pages, they are cached, and some elements might not be recorded in JMeter. I ensure to handle caching issues appropriately to capture all necessary traffic.
- As a result, I obtain a detailed capture of my browsing steps, ready for further analysis or performance testing in JMeter.

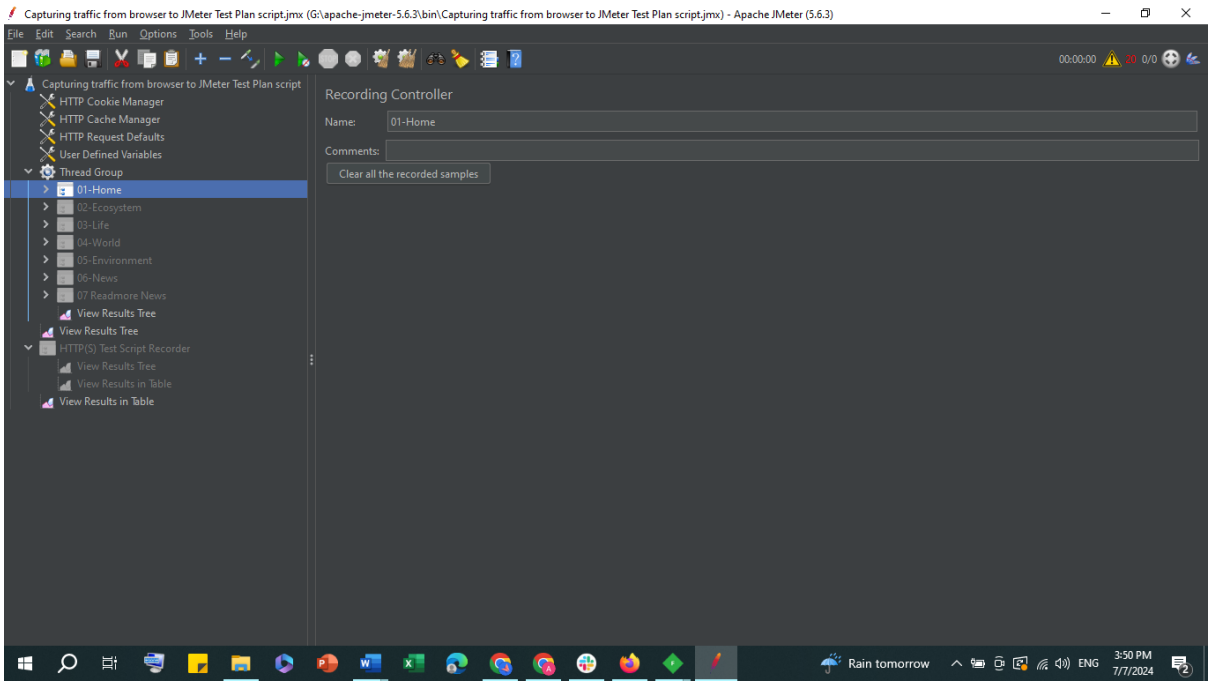
This method allows me to capture accurate traffic flows for each step, aligning with the structured approach necessary for thorough testing.

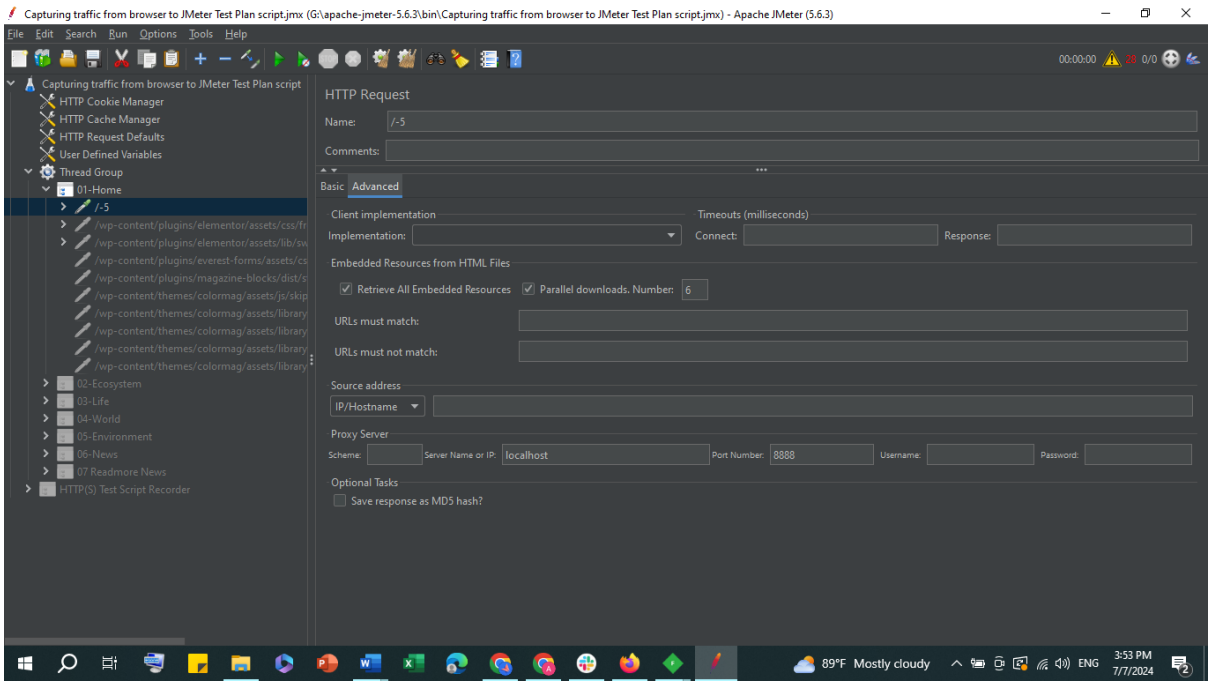
Compare JMeter and browser traffic

To compare JMeter and browser traffic, I use Fiddler as a proxy between them. In the section 'Capturing traffic from browser to Fiddler,' I recorded traffic from each step of the script. Now, to get traffic from JMeter, I need to proxy the traffic from JMeter to Fiddler.

Configuring Proxy in JMeter

1. I go to the 'HTTP Request Defaults' element in JMeter and navigate to the 'Advanced' tab.
2. In the 'Proxy Server' section, I set the following values:
3. Server Name or IP: `localhost`
4. Port Number: `8888` (as in Fiddler)
5. I make sure the checkboxes next to 'Retrieve all Embedded Resources' and 'Parallel downloads' are checked.
6. With these settings, all traffic in JMeter, including embedded resources inside the main request, will be displayed.



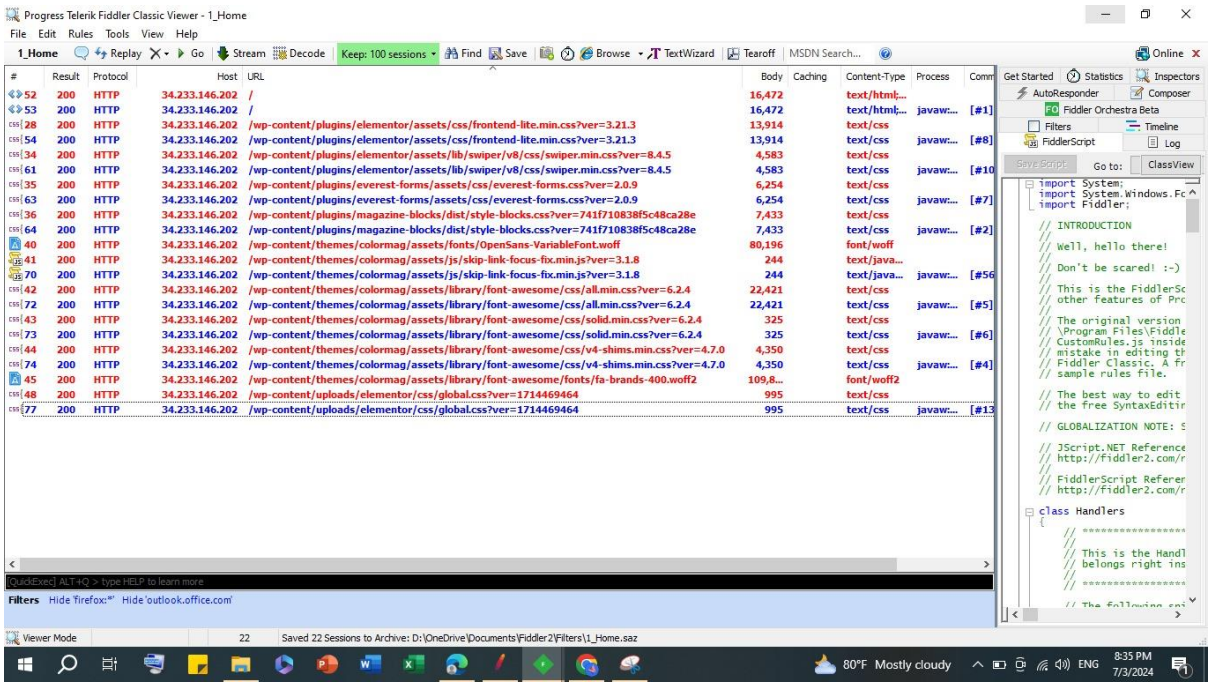


Comparison of the First Step (Open Main Page)

1. **Open the First Transaction and Delete Third-Party Requests Manually**
   - I navigate to the first transaction in Fiddler and manually delete all third-party requests that are not necessary for the comparison.
2. **Check the Embedded Resources from the Main Request**
   - The main request is the first request in the transaction and is usually the URL displayed in the address bar when a user interacts with the page.
   - I disable all requests in the transaction except for the main request.
3. **Disable Other Transactions in the Thread Group**
   - I ensure that only the first transaction is active by disabling the other transactions in the Thread Group.
4. **Start the Thread Group with the First Transaction in JMeter and Check Fiddler**
   - I start the Thread Group with the first transaction in JMeter.
   - I switch to Fiddler to ensure that I see traffic from my host (http://34.233.146.202) inside the first Transaction Controller. If everything is configured correctly, Fiddler will display several requests: the main request and embedded requests.
5. **Open a New Viewer in Fiddler**
   - I navigate to File -> New Viewer in Fiddler to open a new viewer.
   - I load the traffic for the first step from the browser to Fiddler into the new viewer and highlight those requests with a color (e.g., Ctrl + 1 for red).
6. **Highlight Requests from JMeter to Fiddler in a Different Color**
   - I highlight the requests from JMeter to Fiddler with a different color (e.g., Ctrl + 2 for blue).
7. **Allocate All Traffic from JMeter to Fiddler**
   - I select all traffic from JMeter to Fiddler (Ctrl + A) and drag it to the new viewer with traffic from the browser to Fiddler.
8. **Filter Requests by URL**
   - I click on the URL column to filter the requests by their URLs.

- The screenshot below shows what the traffic will look like after using the filter. Some requests are the same, indicating they are inside the main page and do not need to be added to JMeter separately. However, there are requests highlighted in red (from the browser) that do not match the blue requests (from JMeter).

9. **Add Missing Requests to JMeter**
    - This indicates that I need to add the missing requests to JMeter. If these requests do not appear in the transaction, I add them manually. If they appear in the recording, I separate them in the transaction, and all other requests can be deleted manually from the transaction.

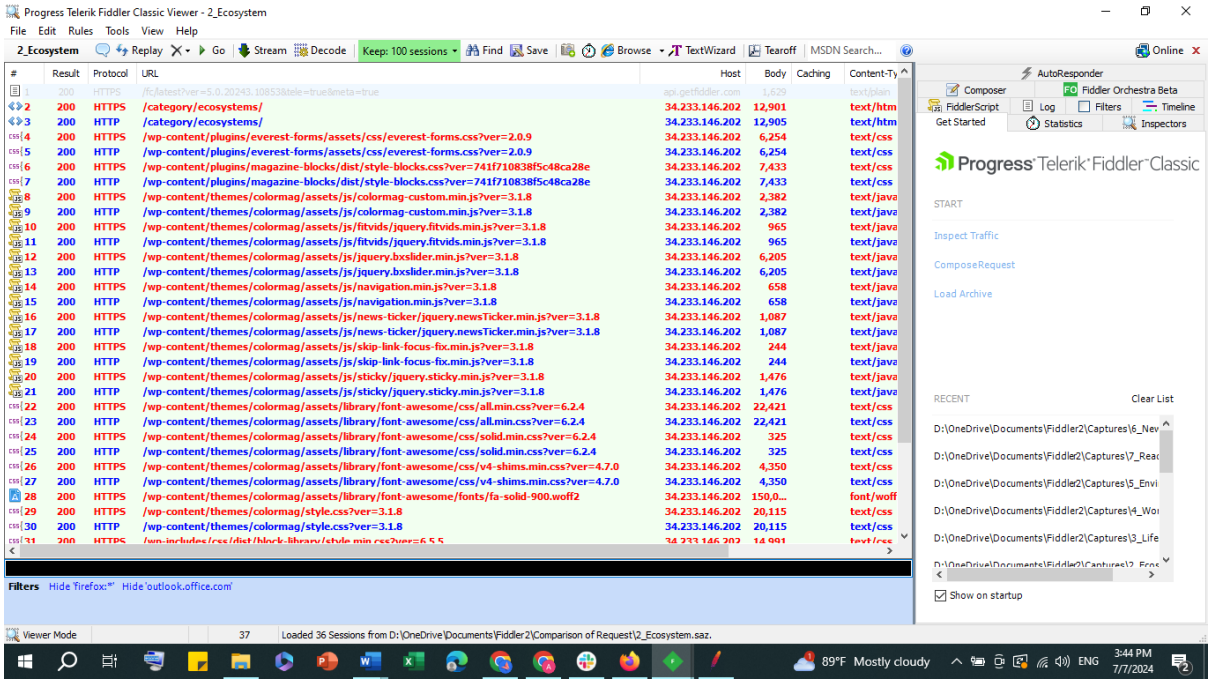


Comparison of the Second Step (Open Ecosystems Page)

1. **Disable Only the Third Transaction in JMeter**
    - I disable only the third transaction in JMeter, keeping the first and second transactions active.
2. **Prepare Fiddler for Capturing Traffic from JMeter**
    - I ensure Fiddler is ready to capture traffic from JMeter.
3. **Disable All Requests in the Second Transaction Except the Main Request**
    - I disable all requests in the second transaction except for the main request (first request).
4. **Start the Thread Group with the Second Transaction in JMeter and Check Fiddler**
    - I start the Thread Group with the second transaction in JMeter.
    - I switch to Fiddler to ensure that I see traffic from my host (http://34.233.146.202) inside the second Transaction Controller.
5. **Open a New Viewer in Fiddler**
    - I navigate to File -> New Viewer in Fiddler to open a new viewer.
    - I load the traffic for the second step from the browser to Fiddler into the new viewer and highlight those requests with a color (e.g., Ctrl + 1 for red).
6. **Highlight Requests from JMeter to Fiddler in a Different Color**

- I highlight the requests from JMeter to Fiddler with a different color (e.g., Ctrl + 2 for blue).

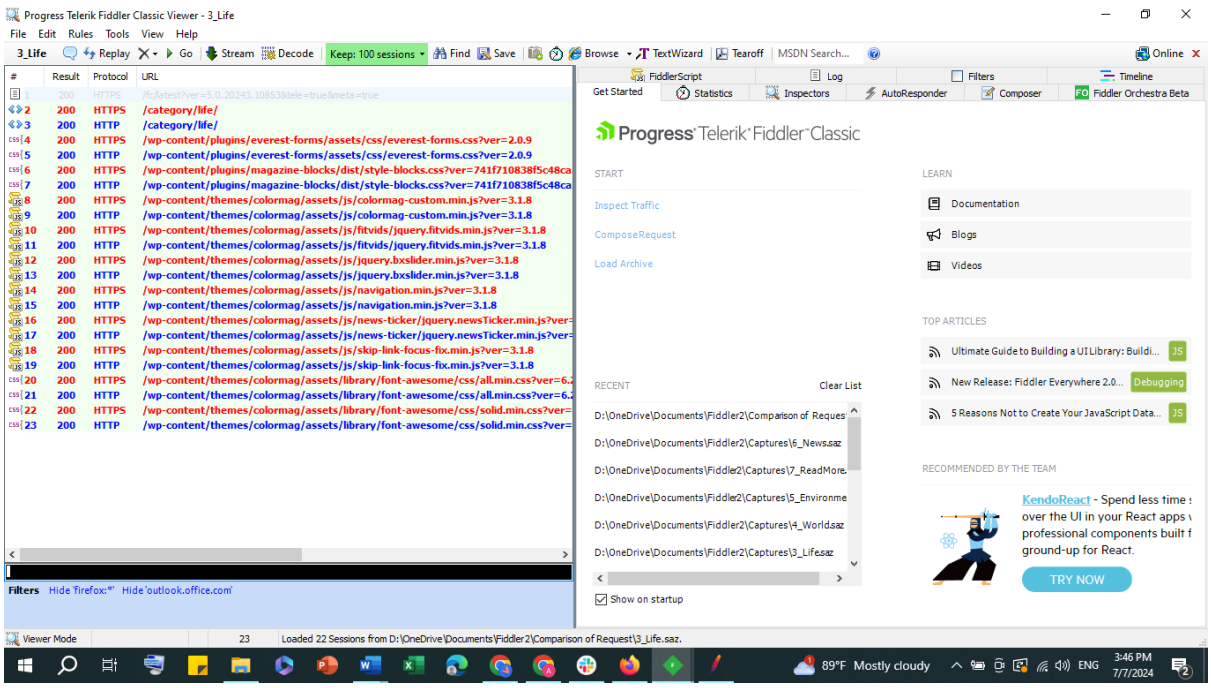
7. **Allocate All Traffic from JMeter to Fiddler**
   - I select all traffic from JMeter to Fiddler (Ctrl + A) and drag it to the new viewer with traffic from the browser to Fiddler.
8. **Filter Requests by URL**
   - I click on the URL column to filter the requests by their URLs.
   - I compare the requests highlighted in red (from the browser) and blue (from JMeter).
9. **Add Missing Requests to JMeter**
   - If any requests from the browser traffic (red) do not match those from JMeter (blue), I add them manually to the second transaction in JMeter.



Comparison of the Third Step (Open Life Page)

1. **Enable All Transactions in JMeter**
   - I enable all transactions in JMeter to prepare for the third comparison.
2. **Prepare Fiddler for Capturing Traffic from JMeter**
   - I ensure Fiddler is ready to capture traffic from JMeter.
3. **Disable All Requests in the Third Transaction Except the Main Request**
   - I disable all requests in the third transaction except for the main request (first request).
4. **Start the Thread Group with the Third Transaction in JMeter and Check Fiddler**
   - I start the Thread Group with the third transaction in JMeter.
   - I switch to Fiddler to ensure that I see traffic from my host (http://34.233.146.202) inside the third Transaction Controller.
5. **Open a New Viewer in Fiddler**
   - I navigate to File -> New Viewer in Fiddler to open a new viewer.
   - I load the traffic for the third step from the browser to Fiddler into the new viewer and highlight those requests with a color (e.g., Ctrl + 1 for red).

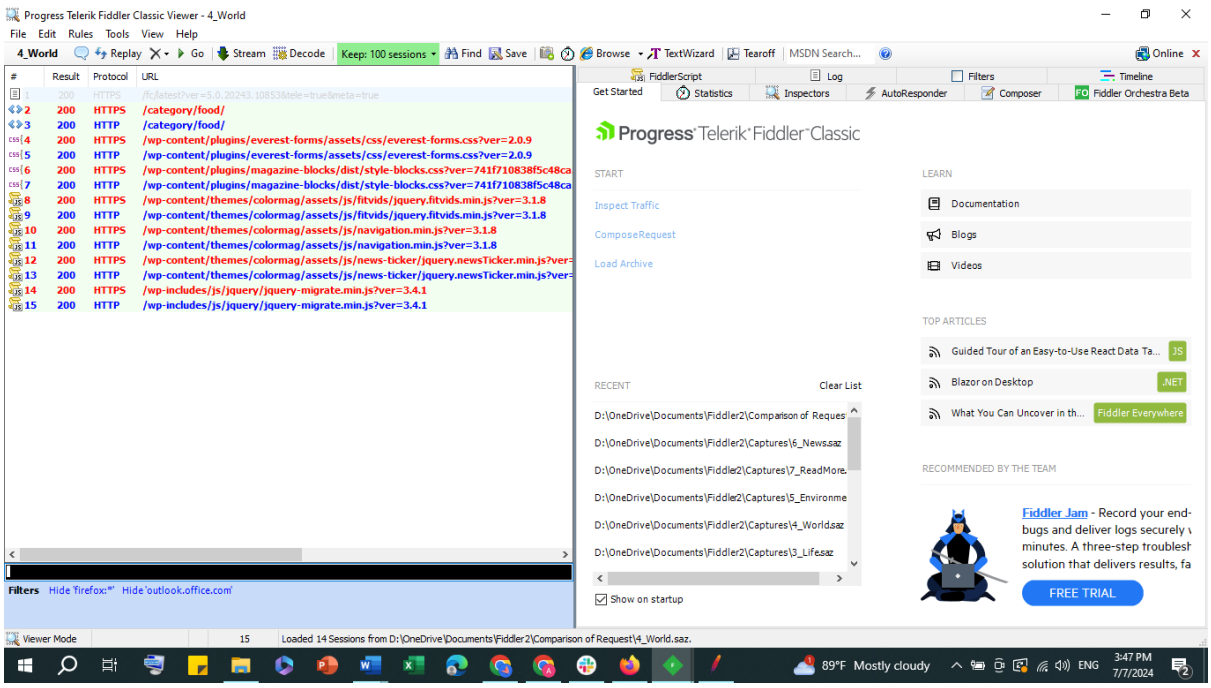
6. **Highlight Requests from JMeter to Fiddler in a Different Color**
    - I highlight the requests from JMeter to Fiddler with a different color (e.g., Ctrl + 2 for blue).
7. **Allocate All Traffic from JMeter to Fiddler**
    - I select all traffic from JMeter to Fiddler (Ctrl + A) and drag it to the new viewer with traffic from the browser to Fiddler.
8. **Filter Requests by URL**
    - I click on the URL column to filter the requests by their URLs.
    - I compare the requests highlighted in red (from the browser) and blue (from JMeter).
9. **Add Missing Requests to JMeter**
    - If any requests from the browser traffic (red) do not match those from JMeter (blue), I add them manually to the third transaction in JMeter.



Comparison of the Fourth Step (Open World Page)

1. **Disable All Transactions Except the Fourth in JMeter**
    - I disable all transactions except for the fourth one in JMeter.
2. **Prepare Fiddler for Capturing Traffic from JMeter**
    - I ensure Fiddler is ready to capture traffic from JMeter.
3. **Disable All Requests in the Fourth Transaction Except the Main Request**
    - I disable all requests in the fourth transaction except for the main request (first request).
4. **Start the Thread Group with the Fourth Transaction in JMeter and Check Fiddler**
    - I start the Thread Group with the fourth transaction in JMeter.
    - I switch to Fiddler to ensure that I see traffic from my host (http://34.233.146.202) inside the fourth Transaction Controller.
5. **Open a New Viewer in Fiddler**
    - I navigate to File -> New Viewer in Fiddler to open a new viewer.
    - I load the traffic for the fourth step from the browser to Fiddler into the new viewer and highlight those requests with a color (e.g., Ctrl + 1 for red).

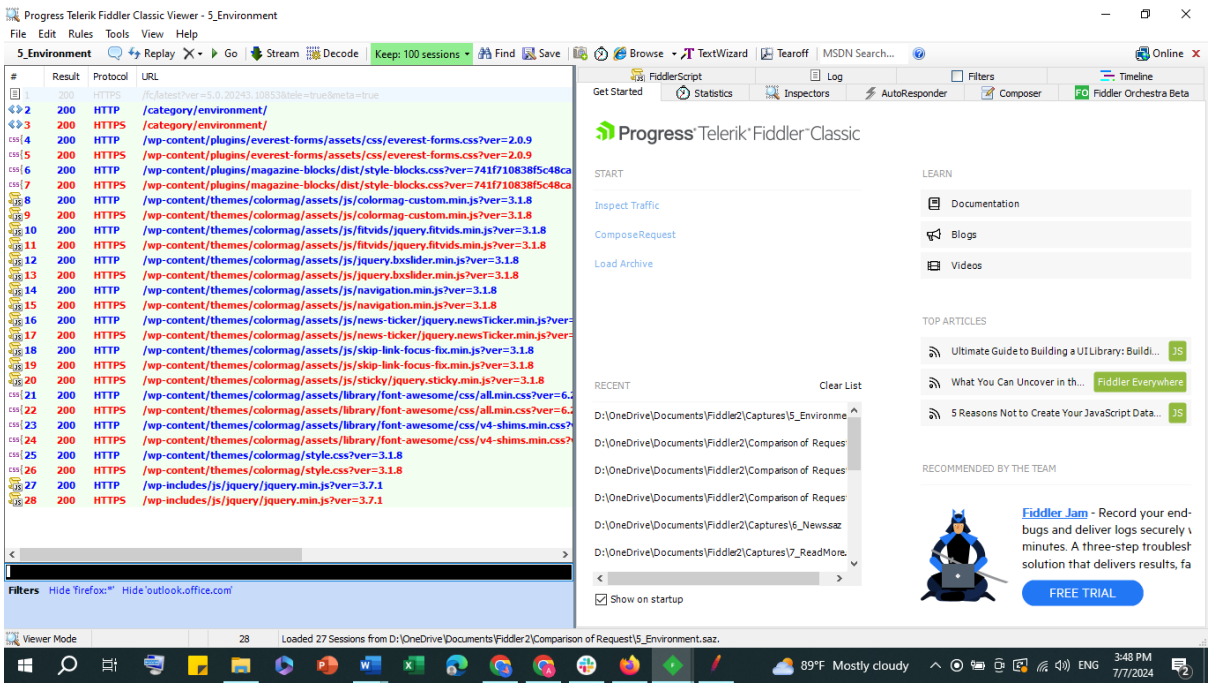
6. **Highlight Requests from JMeter to Fiddler in a Different Color**
    - I highlight the requests from JMeter to Fiddler with a different color (e.g., Ctrl + 2 for blue).
7. **Allocate All Traffic from JMeter to Fiddler**
    - I select all traffic from JMeter to Fiddler (Ctrl + A) and drag it to the new viewer with traffic from the browser to Fiddler.
8. **Filter Requests by URL**
    - I click on the URL column to filter the requests by their URLs.
    - I compare the requests highlighted in red (from the browser) and blue (from JMeter).
9. **Add Missing Requests to JMeter**
    - If any requests from the browser traffic (red) do not match those from JMeter (blue), I add them manually to the fourth transaction in JMeter.



Comparison of the Fifth Step (Open Environment Page)

1. **Disable All Transactions Except the Fifth in JMeter**
    - I disable all transactions except for the fifth one in JMeter.
2. **Prepare Fiddler for Capturing Traffic from JMeter**
    - I ensure Fiddler is ready to capture traffic from JMeter.
3. **Disable All Requests in the Fifth Transaction Except the Main Request**
    - I disable all requests in the fifth transaction except for the main request (first request).
4. **Start the Thread Group with the Fifth Transaction in JMeter and Check Fiddler**
    - I start the Thread Group with the fifth transaction in JMeter.
    - I switch to Fiddler to ensure that I see traffic from my host (http://34.233.146.202) inside the fifth Transaction Controller.
5. **Open a New Viewer in Fiddler**
    - I navigate to File -> New Viewer in Fiddler to open a new viewer.
    - I load the traffic for the fifth step from the browser to Fiddler into the new viewer and highlight those requests with a color (e.g., Ctrl + 1 for red).

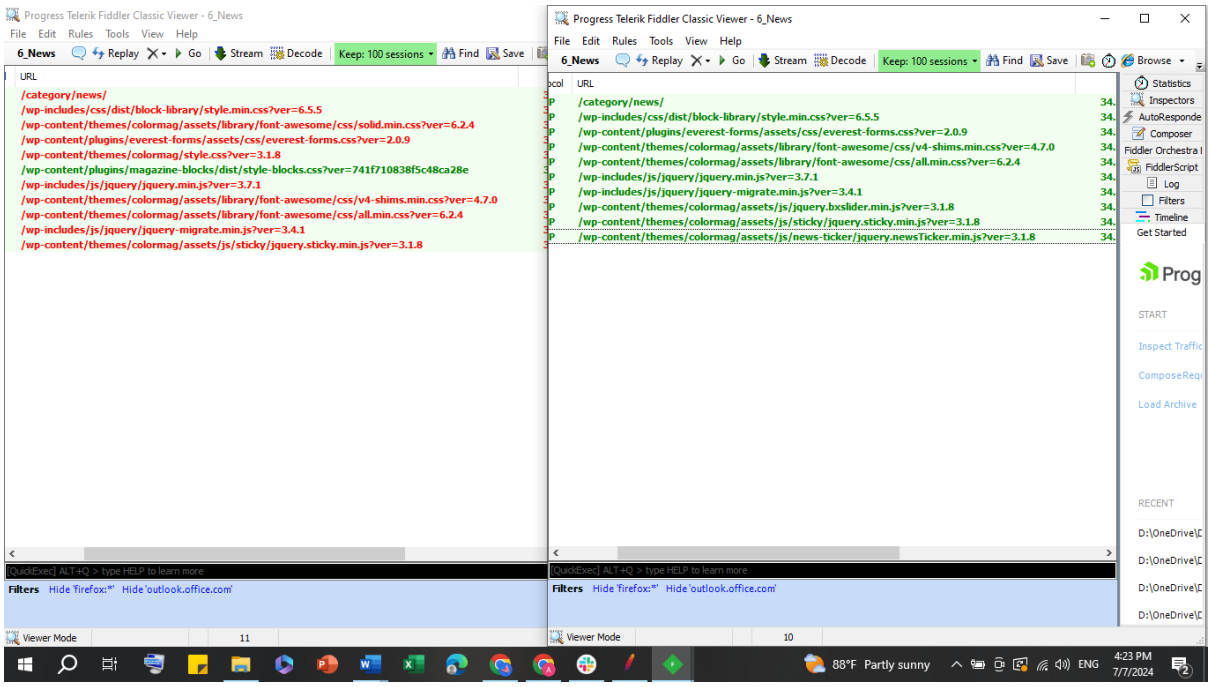
6. **Highlight Requests from JMeter to Fiddler in a Different Color**
    - I highlight the requests from JMeter to Fiddler with a different color (e.g., Ctrl + 2 for blue).
7. **Allocate All Traffic from JMeter to Fiddler**
    - I select all traffic from JMeter to Fiddler (Ctrl + A) and drag it to the new viewer with traffic from the browser to Fiddler.
8. **Filter Requests by URL**
    - I click on the URL column to filter the requests by their URLs.
    - I compare the requests highlighted in red (from the browser) and blue (from JMeter).
9. **Add Missing Requests to JMeter**
    - If any requests from the browser traffic (red) do not match those from JMeter (blue), I add them manually to the fifth transaction in JMeter.

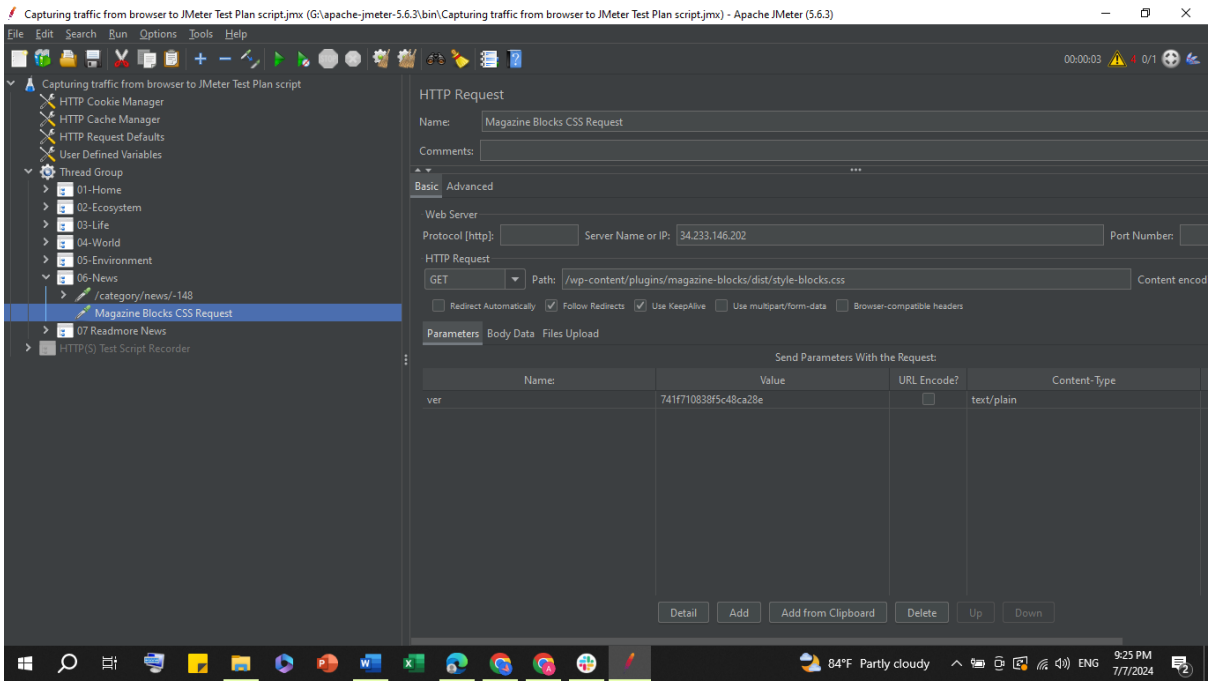


Comparison of the Sixth Step (Open News Page)

1. **Disable All Transactions Except the Sixth in JMeter**
    - I disable all transactions except for the sixth one in JMeter.
2. **Prepare Fiddler for Capturing Traffic from JMeter**
    - I ensure Fiddler is ready to capture traffic from JMeter.
3. **Disable All Requests in the Sixth Transaction Except the Main Request**
    - I disable all requests in the sixth transaction except for the main request (first request).
4. **Start the Thread Group with the Sixth Transaction in JMeter and Check Fiddler**
    - I start the Thread Group with the sixth transaction in JMeter.
    - I switch to Fiddler to ensure that I see traffic from my host (http://34.233.146.202) inside the sixth Transaction Controller.
5. **Open a New Viewer in Fiddler**
    - I navigate to File -> New Viewer in Fiddler to open a new viewer.
    - I load the traffic for the sixth step from the browser to Fiddler into the new viewer and highlight those requests with a color (e.g., Ctrl + 1 for red).

6. **Highlight Requests from JMeter to Fiddler in a Different Color**
    - I highlight the requests from JMeter to Fiddler with a different color (e.g., Ctrl + 2 for blue).
7. **Allocate All Traffic from JMeter to Fiddler**
    - I select all traffic from JMeter to Fiddler (Ctrl + A) and drag it to the new viewer with traffic from the browser to Fiddler.
    - Here I was facing that the drag option was not working, so I put individual windows of Fiddler and JMeter captured traffic together and compared the requests.
8. **Filter Requests by URL**
    - I click on the URL column to filter the requests by their URLs.
    - I compare the requests highlighted in red (from the browser) and blue (from JMeter).
9. **Add Missing Requests to JMeter**
    - If any requests from the browser traffic (red) do not match those from JMeter (blue), I add them manually to the sixth transaction in JMeter.

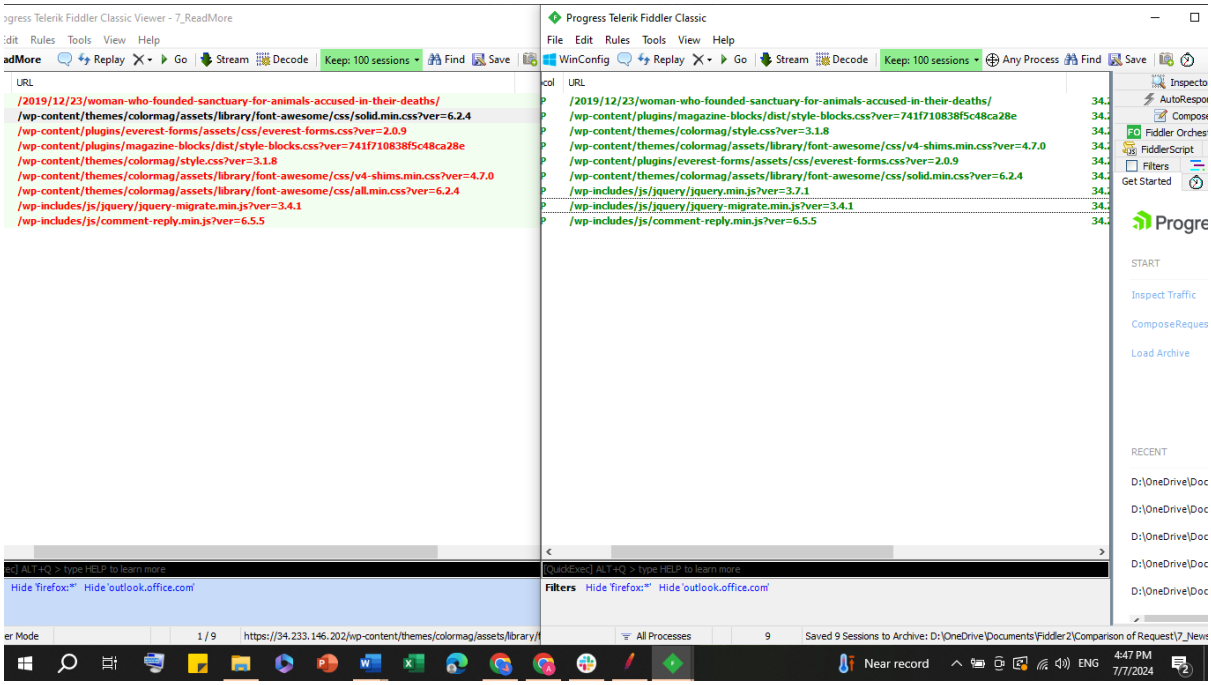




Comparison of the Seventh Step (Read More on News Page)

1. **Disable All Transactions Except the Seventh in JMeter**
   - I disable all transactions except for the seventh one in JMeter.
2. **Prepare Fiddler for Capturing Traffic from JMeter**
   - I ensure Fiddler is ready to capture traffic from JMeter.
3. **Disable All Requests in the Seventh Transaction Except the Main Request**
   - I disable all requests in the seventh transaction except for the main request (first request).
4. **Start the Thread Group with the Seventh Transaction in JMeter and Check Fiddler**
   - I start the Thread Group with the seventh transaction in JMeter.
   - I switch to Fiddler to ensure that I see traffic from my host (http://34.233.146.202) inside the seventh Transaction Controller.
5. **Open a New Viewer in Fiddler**
   - I navigate to File -> New Viewer in Fiddler to open a new viewer.
   - I load the traffic for the seventh step from the browser to Fiddler into the new viewer and highlight those requests with a color (e.g., Ctrl + 1 for red).
6. **Highlight Requests from JMeter to Fiddler in a Different Color**
   - I highlight the requests from JMeter to Fiddler with a different color (e.g., Ctrl + 2 for blue).
7. **Allocate All Traffic from JMeter to Fiddler**
   - I select all traffic from JMeter to Fiddler (Ctrl + A) and drag it to the new viewer with traffic from the browser to Fiddler.
   - Here I was facing that the drag option was not working, so I put individual windows of Fiddler and JMeter captured traffic together and compared the requests.

8. **Filter Requests by URL**
    - I click on the URL column to filter the requests by their URLs.
    - I compare the requests highlighted in red (from the browser) and blue (from JMeter).
9. **Add Missing Requests to JMeter**
    - If any requests from the browser traffic (red) do not match those from JMeter (blue), I add them manually to the seventh transaction in JMeter.



Summary

By meticulously comparing each step's traffic between JMeter and the browser, I ensure that JMeter's recorded traffic accurately mirrors real user interactions. This thorough approach allows for precise performance testing and validation, leading to reliable results and insights.

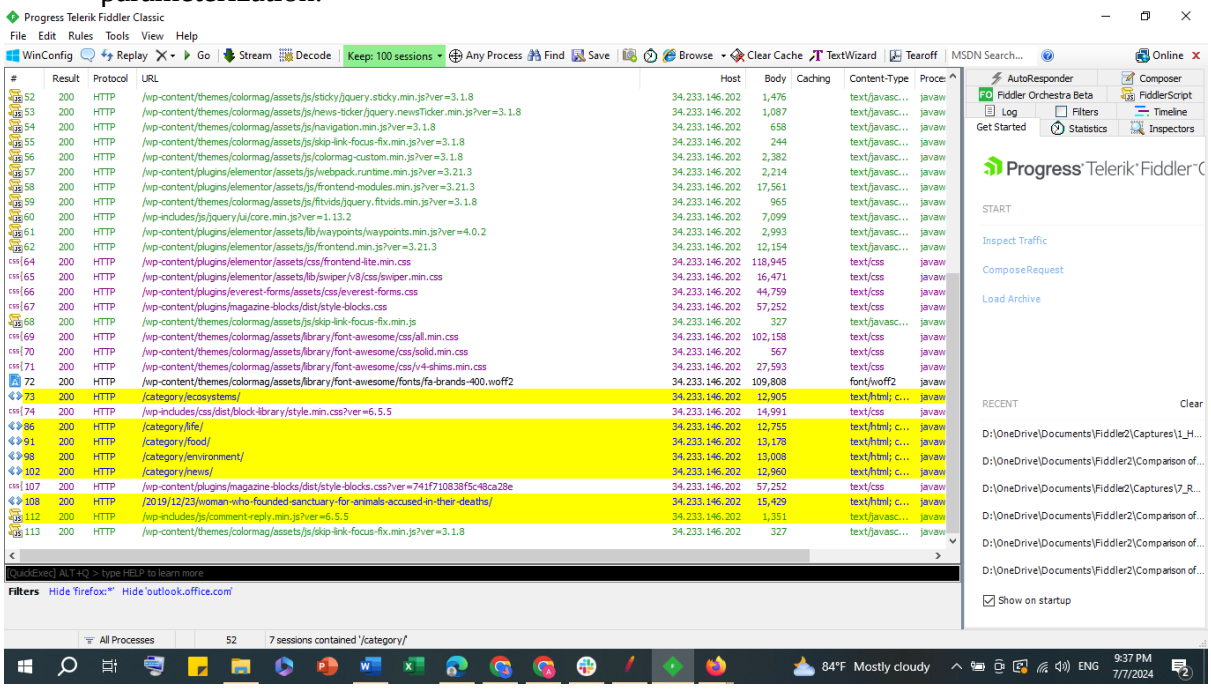
Value parametrization

To open a random category each time in my script, I need to create a regular expression that extracts links to all catalogs on the page and parameterizes them in the request. Here's a detailed guide on how to do this:

➢ **Step 1: Inspect Traffic in Fiddler**

1. **Open Fiddler and Navigate to the Traffic from the First and Second Steps**
    - Launch Fiddler and ensure that it is capturing traffic from my IP address `http://34.233.146.202`.
    - Use the search function in Fiddler (Ctrl + F) to find matches for the URL

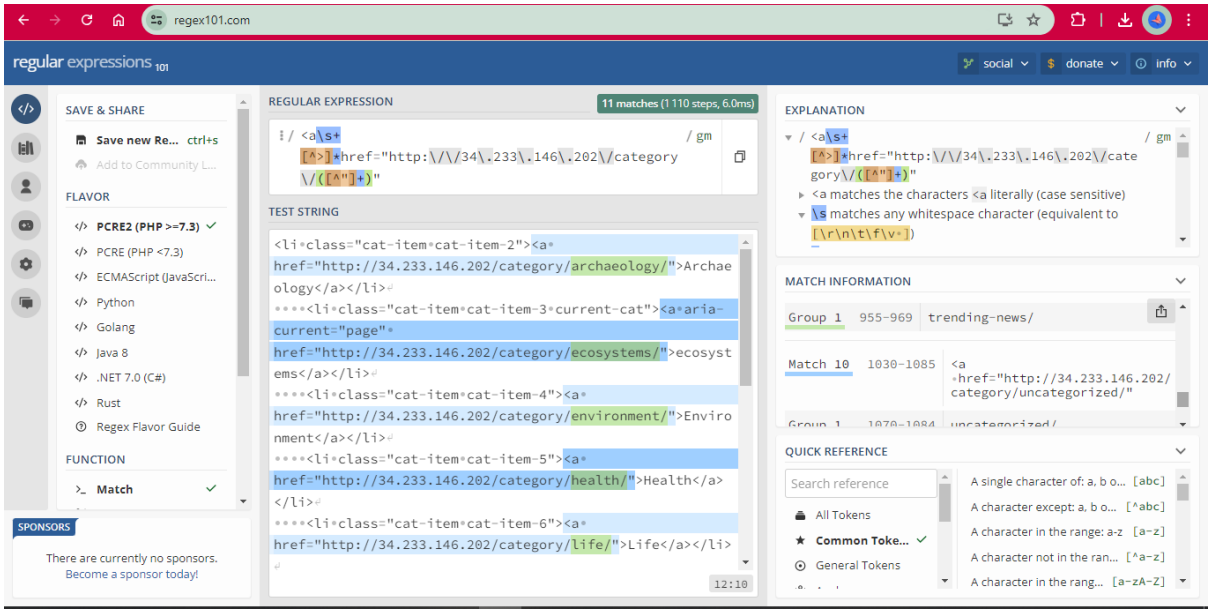
2. **Identify Relevant Requests**
    - In the Fiddler window, requests matching `/category/` are highlighted in yellow. This indicates that these requests contain the URLs I need for parameterization.



3. **Copy the Response from the Request**
    - Right-click on the request with the matching URL and select ‘Inspect’ to view the response details.
    - Copy the response body, which contains the links to all categories on the page.

➢ **Step 2: Create the Regular Expression**

1. **Navigate to https://regex101.com/**
    - Open your web browser and go to Regex101.
2. **Paste the Response into Regex101**
    - Paste the copied response into the input field on Regex101.
3. **Construct the Regular Expression**
    - Use Regex101’s tools to construct a regular expression that matches all category links. For instance, you might use a pattern like:
    - `<a\s+[^>]*href="http:\/\/34\.233\.146\.202\/category\/([^"]+)"`



➢ **Step 3: Add the Regular Expression Extractor in JMeter**

1. **Open JMeter and Navigate to Your Test Plan**
   - Launch JMeter and open the test plan where you want to add the parameterization.
2. **Add the Regular Expression Extractor**
   - Select the HTTP Request element (`/` request).
   - Go to `Request -> Add -> Post Processors -> Regular Expression Extractor`.

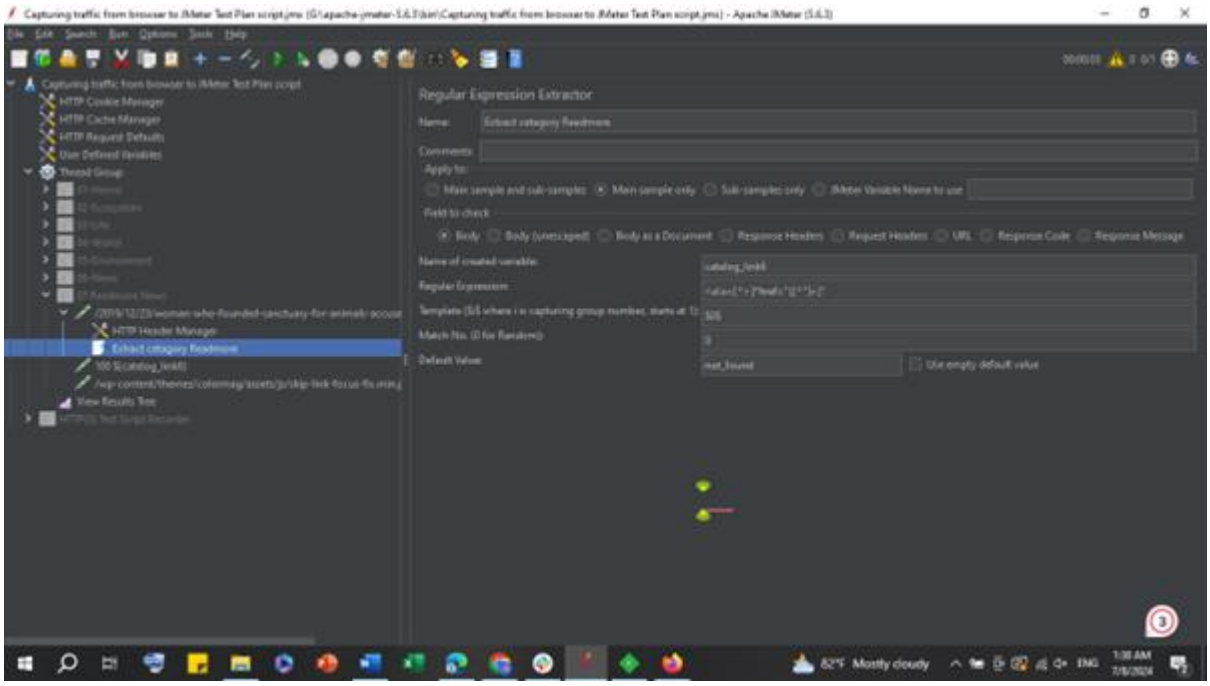
➢ **Step 4: Parameterize the URL in the HTTP Request**

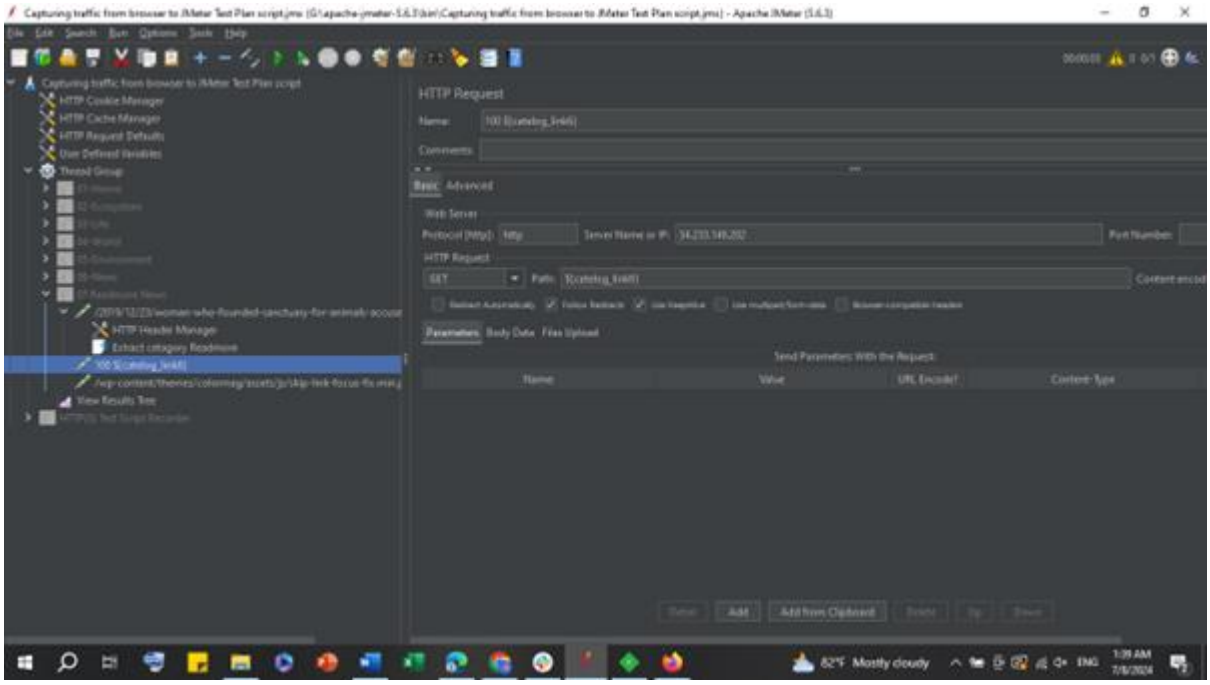
1. **Edit the HTTP Request Element**
   - Go to the HTTP Request element where you want to parameterize the URL.
2. **Modify the URL to Use the Extracted Value**
   - Use the `${catelog_link6}` variable in the URL. For example:
3. **Example URL Configuration**
   - Your HTTP Request URL might look like this:

```
http://34.233.146.202/category/${catelog_link6}
```

➢ **Step 5: Test and Validate**

1. **Run the Test Plan in JMeter**
   - Start the test plan execution in JMeter.





2. **Monitor Results in View Results Tree**
    - Use the ‘View Results Tree’ listener to check if the requests are correctly parameterized.
    - Ensure that each request is sending a different category URL.
3. **Verify Requests in Fiddler**
    - Check Fiddler to ensure that the requests sent by JMeter include the parameterized category URLs.

Conclusion

By following these detailed steps, I can successfully parameterize the category URLs in JMeter, enabling the script to open a random category each time. This approach ensures that my tests are dynamic and cover a wide range of scenarios.

Regular expression extractor in JMeter

1. **Apply to (Radio Button Options):**
   - I choose "Main sample and sub-samples" because I want the regular expression to apply to both the main sample and any sub-samples or embedded resources.
2. **Field to Check (Radio Button Options):**
   - I select "Body" because I want to apply the regular expression to the body of the response. This includes the content of the web page, excluding headers.
3. **Name of Created Variable:**
   - I name the variable as `${catelog_link6}` because this variable will store the parsed results from the regular expression.
4. **Regular Expression:**
   - The regular expression I use is `<a\s+[^>]*href="([^"]+)"` This pattern captures the category IDs from URLs like `http://34.233.146.202/2019/12/19/top-10-hotels/`, extracting `category->News->read more` as the category ID.
5. **Template:**
   - I set the template to `$0$` to extract from entire expression.
6. **Match No:**
   - I set this field to `0` because I want to capture random value from the matches.
7. **Default Value:**
   - For the default value, I set it to `not_found`. If the regular expression does not find any matches in the response, JMeter will use this default value.

In my test plan, I apply these configurations to an HTTP Request that fetches URLs dynamically based on the extracted category IDs. This setup ensures that each script execution selects a different category, making the test dynamic and representative of real user interactions.

By meticulously configuring the Regular Expression Extractor in JMeter, I ensure that my performance tests accurately simulate user behavior across various categories on the website. This approach enhances the robustness of my testing methodology, providing reliable performance metrics and insights.