ResultHandler-LiveDiff

In Galore, one of the irritating issue is how we can display the vast data of differences(from capture responses to replay responses). This document only covers the result handler for http protocols. The other protocols can have their own specialization, but that would be out of the scope for this document.
some of the reasons why this is difficult

- each page can have multiple requests, displaying them as request response pair could be difficult for the user to comprehend.
- front end java scripts can do lot of magic that is also hard to show the effects as is
- Following diff modes are being considered; however they all have their own advantages and disadvantages
    - side by side diff (this is regular way of showing diffs, like any file diff program)
    - in-line diff (showing the diffs as a single page, with common data displayed only once and the diffs are shown in different colors)

Live Diff

It is obvious that the main objective of the result handler is to display the results in way that is easily understandable to the user. This can only be achieved if we show the whole web page as is. Here is how we are going to recreate the page.

- The first http GET request fetches the whole html text. However, based on how many images/css etc., the browser is going to make further requests.
- we run a local server(our own) on a port that acts as a fake web server to cater the requests from the browser. This is going to fool the web browser into believing that it is connected to the original web server.
- As we capture every request and response at the server, we can see the first response going from the server. We can parse the original response to see all the places in the html that makes the browser send the further requests.
- Now we can do a little bit magic here; we can read ahead all the further requests and change the original html response in such a way that the further request that browser is going to make should instead point to the local web server(which already has the ready response read from the database).
- This way the browser can load the whole page as is with the java script effects happening live. we can slow the loading(user configured settings) and the users can see the diff as it is happening.
- user can pause it, replay it the way he/she wants
- This way the user can see, what is building up before the diff.

.

Java Script

Handling java script is not as simple as described above. Here are some things that we should consider

- we only capture request/responses in the networks and we do not have any way to know what the user clicked on the browser that generates a request. In short, we cannot capture the user interaction with the browser.
- how can we re-create the java scripts effects on the browser?
    - First we preview the html that we want to run in our own browser(which could be a open source browser that has a decent java script engine).
    - after loading the java script, we store the final html generated.
    - for all the components in the web page, we fire events(user interactions), and we store the requests generated.
    - we try to match these requests generated, with the requests that we got during capture/replay to finally identify the components / actions that can possibly generate the requests(during capture/replay).
    - assuming that a unique request is generated for each component/event, this model works.
    - after identifying the components and corresponding events, we can load the page(in the browser that user uses to access crp) and fire the events for which we found the match(in the capture requests)