Artículo
Bernardo Linarez · 30 mar, 2020 Lectura de 6 min

# Depuración web

¡Hola Comunidad!

En este artículo hablaré sobre las pruebas y la depuración de las aplicaciones web de Caché (principalmente REST) con herramientas externas. La segunda parte trata sobre las herramientas de Caché.

Usted escribió el código del lado del servidor y quiere probarlo con un cliente, o ya tiene una aplicación web pero no funciona. Aquí es donde entra la depuración. En este artículo abarcaré desde las herramientas más fáciles de utilizar (el navegador), hasta las más completas (el analizador de paquetes), pero primero conversemos un poco sobre los errores más comunes y cómo pueden resolverse.

## Los errores

### Error 401 No autorizado

Creo que este error es el que encontramos con más frecuencia durante la implementación de la producción. El servidor de desarrollo local normalmente tiene una configuración de seguridad mínima o normal pero muy convencional. Sin embargo, el servidor de producción puede tener un esquema con más restricciones. Entonces:

- Compruebe que ha iniciado sesión
- Compruebe que el usuario tiene acceso a la base de datos/tabla/procedimiento/fila/columna a la que desea acceder
- Compruebe que la solicitud para las OPCIONES puede realizarse por un usuario no autorizado

### Error 404 No encontrado

Compruebe:

- Que la URL es correcta
- Si es una nueva aplicación y está utilizando un servidor web externo, cargar nuevamente el servidor web puede ser útil

### Errores en la aplicación

En cierto modo estos son los más fáciles de encontrar - realizar un seguimiento del *stack* será muy útil. La resolución es totalmente específica para la aplicación.

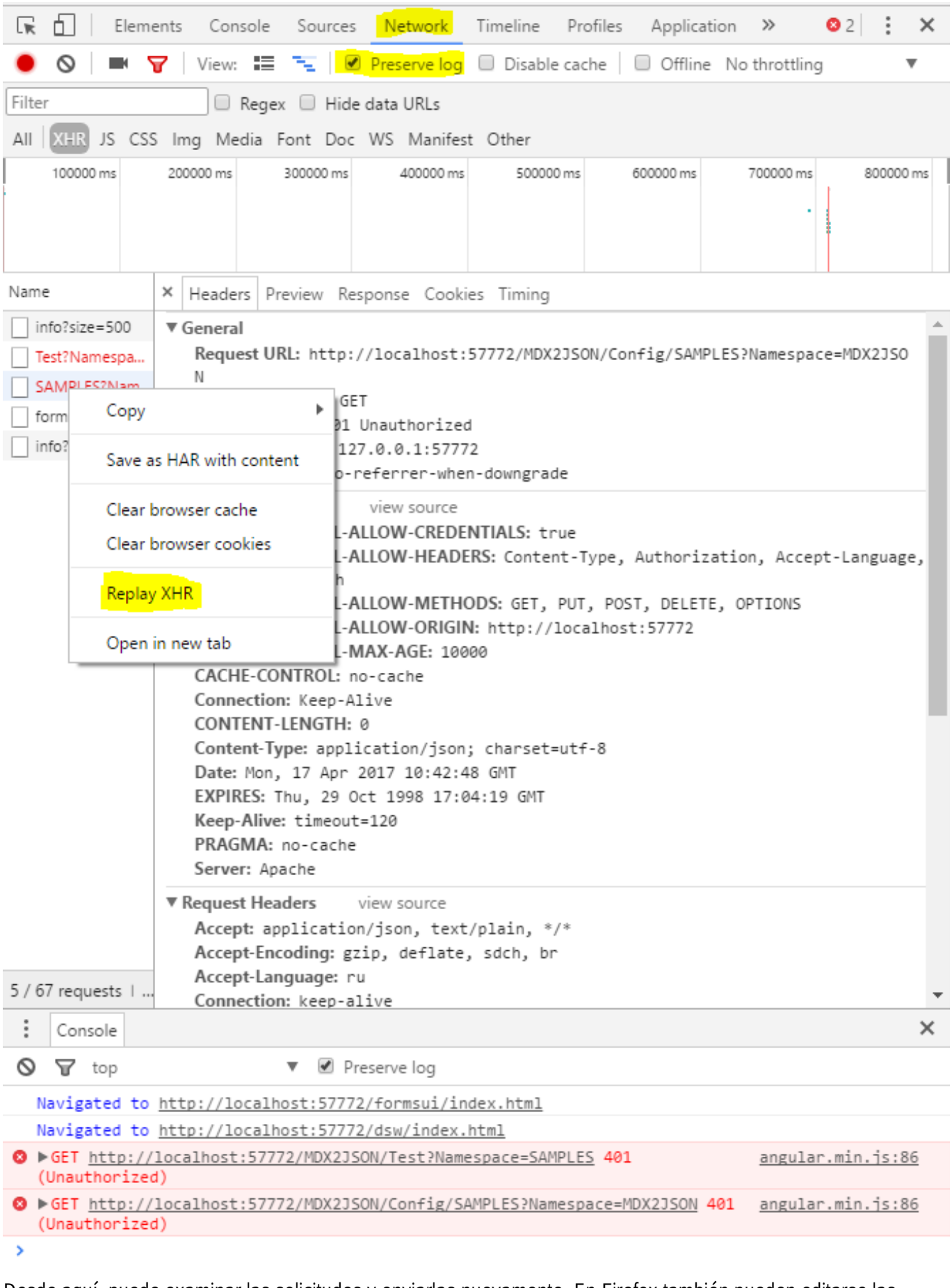
## Herramientas de depuración

### Navegador web

La primera herramienta de depuración que siempre está disponible es un navegador web, es preferible que utilice Chrome, pero Firefox también sería suficiente. Las solicitudes GET pueden probarse al ingresar en la URL que se encuentra en la barra de direcciones, todas las demás solicitudes necesitan de una aplicación web o que estén escritas en un código js. El método general es el siguiente:

- Presione F12 para abrir las herramientas del desarrollador

- Vaya a la pestaña Network
- Marque la casilla Preserve Log, si no está marcada
- Despliegue únicamente las solicitudes XHR
- Realice una acción para depurar errores en la aplicación web




Desde aquí, puede examinar las solicitudes y enviarlas nuevamente. En Firefox también pueden editarse las solicitudes antes de repetirlas.

Ventajas:

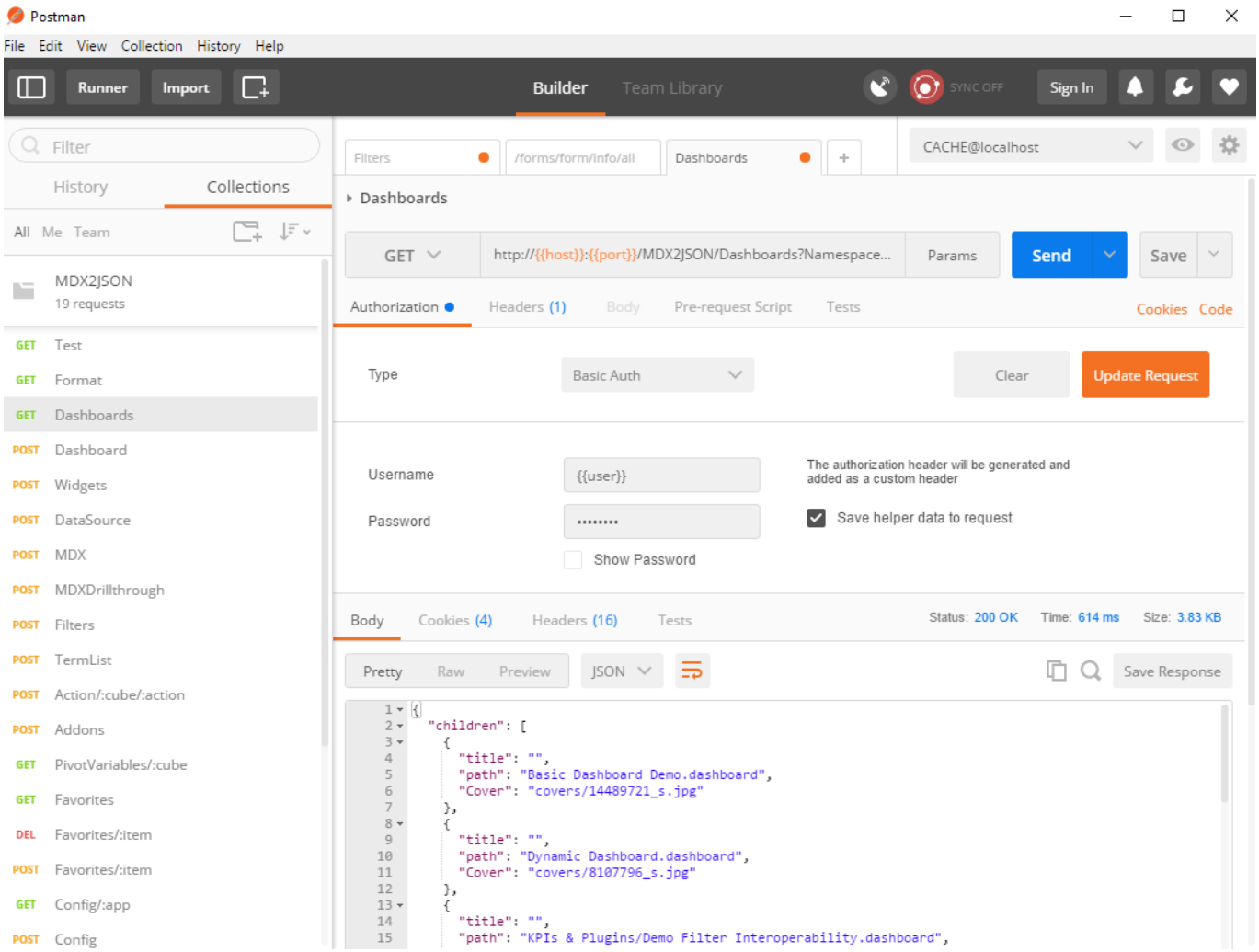
- Siempre está disponible
- Es fácil de utilizar (los usuarios finales pueden enviar capturas de pantalla de la red y las pestañas de la consola)
- Este es el entorno del usuario final

Desventajas:

- No muestra respuestas parciales como send/broken/ etc.
- Es lento con las respuestas que son grandes
- Es lento cuando hay una gran cantidad de respuestas
- Todo se hace manualmente

### Cliente REST

El cliente REST es una aplicación web independiente o un complemento del navegador, que se diseñó específicamente para probar aplicaciones web. Aquí utilicé Postman, pero existen muchos otros. Así es como se ven las depuraciones en Postman:




Postman funciona mediante solicitudes agrupadas en colecciones. La solicitud puede enviarse a un entorno. El entorno es una colección de variables. Por ejemplo, en mi entorno CACHE@localhost la variable del host está configurada como localhost y el usuario como SYSTEM. Cuando se envía una solicitud, las variables se reemplazan por sus valores en el entorno seleccionado y entonces la solicitud se envía.

Aquí puede consultar una recopilación de ejemplos y del entorno para el proyecto MDX2JSON.

Ventajas:

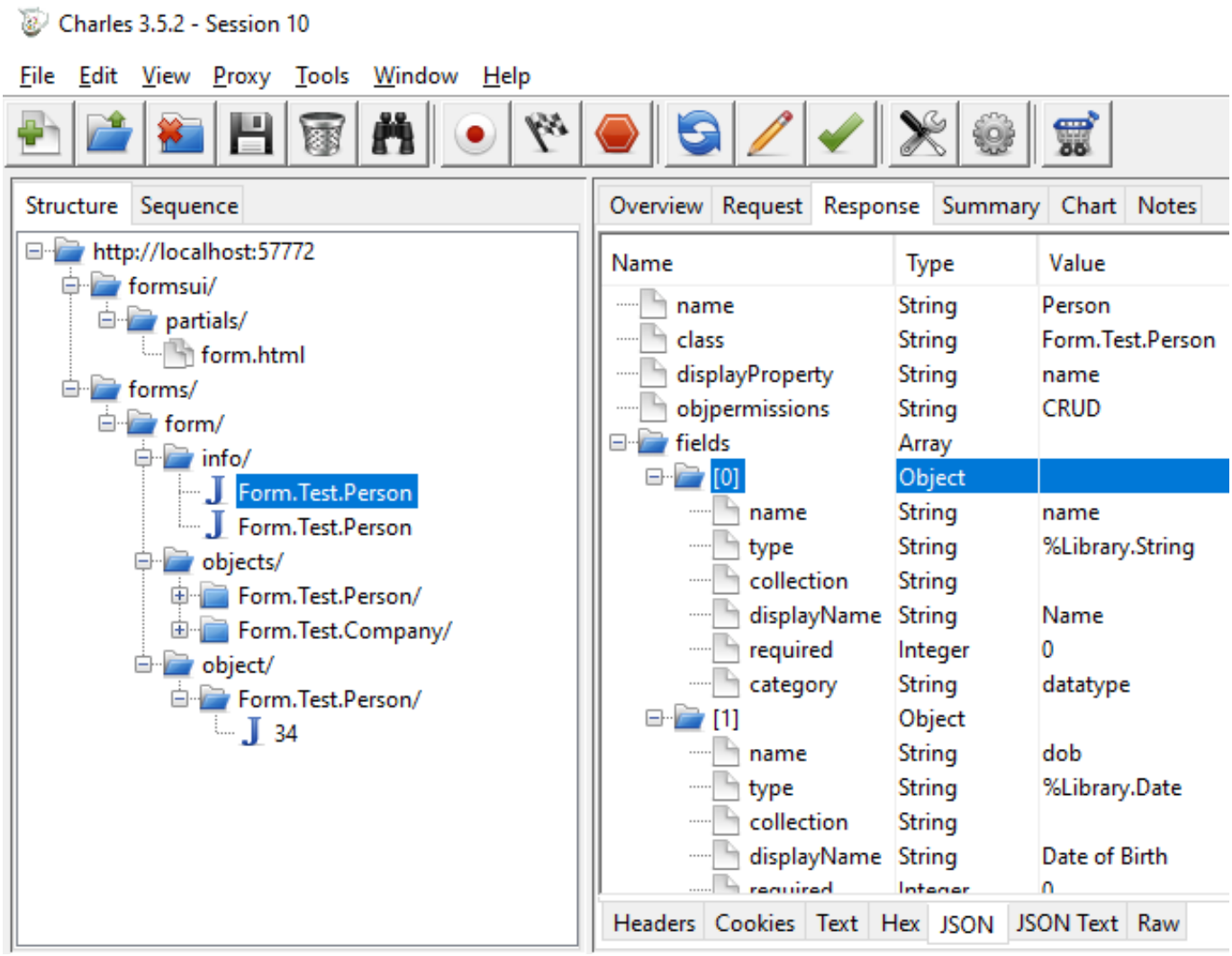
- Solo es necesario escribirlo una vez y podrá utilizarlo en todas partes
- Mejor control de las solicitudes
- Optimización de respuestas

Desventajas:

- La depuración de solicitudes encadenadas sigue siendo manual (la respuesta a la solicitud 1 puede forzar una respuesta, ya sea, en la solicitud 2 o en la solicitud 2B)
- Algunas veces tiene errores en las respuestas parciales como send/broken/etc.

### Proxy de depuraciones HTTP

Es una aplicación independiente que registra el tráfico en los HTTP. Las solicitudes registradas pueden modificarse y reenviarse. Yo utilizo Charles y Fiddler.




Ventajas:

- Procesa respuestas parciales como send/broken/ etc.
- Optimización de respuestas
- Mejor soporte técnico para el tráfico HTTPS (que con el analizador de paquetes)
- Puede guardar sesiones de captura

Desventajas:

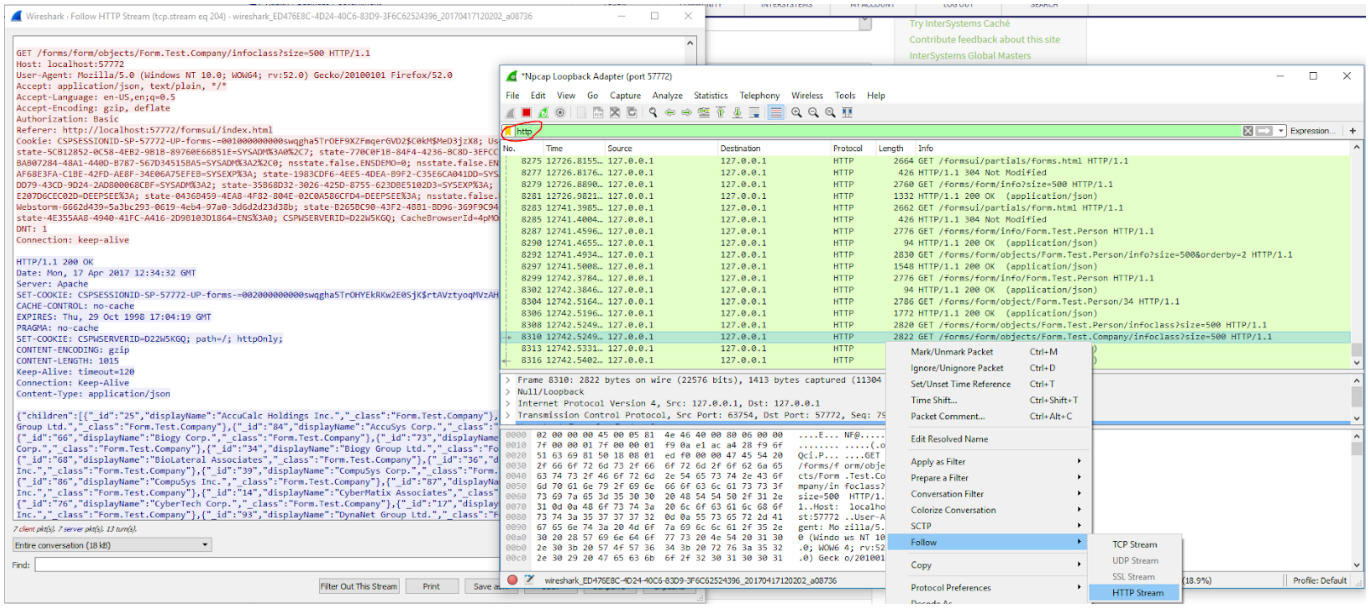
- Algunas cosas son necesarias para enviar la solicitud (por ejemplo, aplicaciones web/cliente REST/código JS)

### Analizador de paquetes

Es un programa computacional que puede interceptar y registrar el tráfico que pasa por una red. Conforme el flujo de datos fluye a través de la red, el rastreador captura cada paquete y, si es necesario, decodifica los datos sin procesar del paquete. Esta es la opción más completa, pero también requiere de ciertas habilidades para funcionar correctamente. Yo utilizo WireShark. Aquí tiene una pequeña guía sobre cómo instalarlo y utilizarlo:

1. Si va a capturar paquetes locales, lea acerca del loopback e instale los requisitos previos del software (npcap para Windows)
2. Instale WireShark
3. Configure los filtros de captura (por ejemplo, un filtro para capturar únicamente el tráfico de la dirección 57772: con el puerto 57772
4. Inicie la captura
5. Configure los filtros de visualización (por ejemplo, un filtro para mostrar únicamente el tráfico http para una IP específica: ip.addr == 1.2.3.4 && http

A continuación, se muestra un ejemplo de captura para el tráfico http (filtro de visualización) en el puerto 57772 (filtro de captura):



Ventajas:

- Procesa respuestas parciales como send/broken/ etc.
- Puede capturar grandes cantidades de tráfico
- Puede capturar cualquier cosa
- Puede guardar sesiones de captura

Desventajas:

- Algunas cosas son necesarias para enviar la solicitud (por ejemplo, aplicaciones web/cliente REST/código JS)

## Qué utilizar

Bueno, eso depende de cuál sea el objetivo. En primer lugar, podemos registrar (depurar el proxy, analizar paquetes) o generar solicitudes (navegadores, clientes REST).

Si está desarrollando una API Web para REST, el cliente REST es la forma más rápida de probar que funciona.

Sin embargo, si las solicitudes del cliente REST funcionan, pero la aplicación web del cliente no, posiblemente necesite un navegador, un proxy de depuraciones http y un analizador de paquetes.

Si tiene clientes y necesita desarrollar una API del lado del servidor para trabajar con ellos, necesitará un proxy de depuraciones http o un analizador de paquetes.

Lo mejor es estar familiarizado con los 4 tipos de herramientas y cambiar rápidamente entre una y otra si la que utiliza actualmente no es suficiente para realizar el trabajo.

Algunas veces resulta evidente cuál es la herramienta correcta.

Por ejemplo, recientemente desarrollé una API en el lado del servidor para un protocolo de extensión http muy popular, los requisitos fueron:

- No podemos cambiar el código que ya habían escrito los clientes
- Clientes diferentes se comportan de distintas maneras
- El comportamiento entre http y https es diferente
- El comportamiento con distintos tipos de autenticación es diferente
- Es posible procesar hasta cien solicitudes por segundo, por cliente
- Todo el mundo ignora el RFC

Aquí solamente existe una solución, el analizador de paquetes.

O si estoy desarrollando una API REST para el consumo de JS, el cliente REST es la herramienta perfecta para realizar pruebas.

Cuando depure la aplicación web, siempre comience con el navegador.

En la parte 2 discutiremos todas las cosas (muchas) que puede hacer para depurar la web en el lado de Caché.

¿Qué métodos utiliza para depurar las comunicaciones entre el cliente y el servidor?

#API REST #CSP #Frontend #Mejores prácticas #SOAP #Caché