

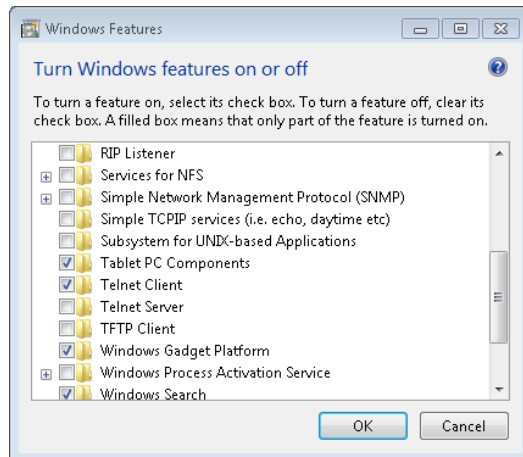
EE4216 Tutorial 1 –Introduction to WWW

Objectives:

- Learn to inspect HTTP requests and responses
 - Discover the syntax of HTML document
 - Emulate an HTTP request using Telnet
 - Learn to use Chrome Developer Tools
 - Understand the application of multi-tier architecture
-
1. We have seen how to use the web-sniffer tool to trace the HTTP messages between browser and web server. Try to use the same tool (<http://websniffer.cc>) to inspect the HTTP request and response **header** to/from CityU's homepage (<https://www.cityu.edu.hk>), and answer the following questions:
 1. What is the IP address of the web server?
 2. What is the referrer? (the referrer header is often used to track the source of the traffic and useful for web analytics)
 3. What is the response status code? What does this code mean?
 4. What is the MIME content type of the response?
 5. What is the size of the response content?
 6. What is the server side programming environment?
 7. Look into the response **body**.
 - A. What is the syntax for comment in HTML?
 - B. What is the syntax for declaring a JavaScript section?
 - C. What is the meaning of the code at first line `<!DOCTYPE html>` ? What is the HTML version of this document?
 8. Select and save the response content to a HTML file, then open it with the browser. Can you see the content? Why are the styles and images missing from this local page?

2. You can see the HTTP header from a web site using the standard Telnet program that comes with Windows. For example, to see the header from the root page of the CityU web site, do these steps:

1. Go to Control Panel > Programs > Turn Windows features on or off. Make sure the Telnet Client program is available.

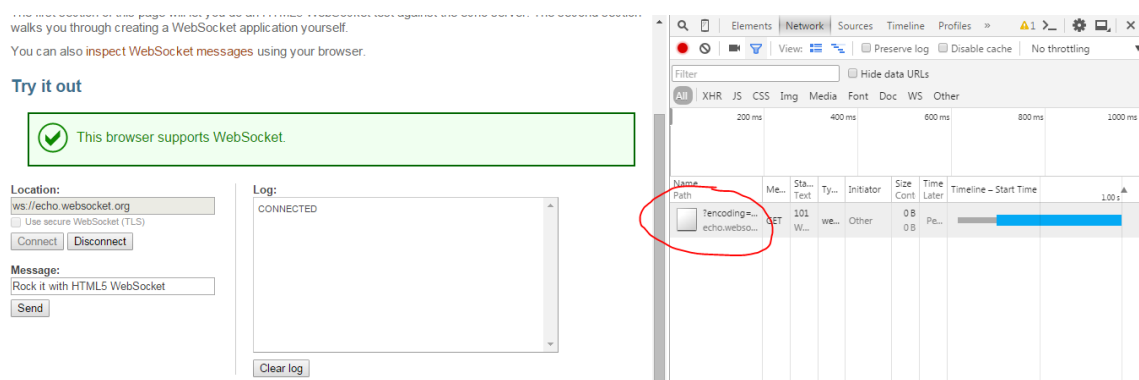


2. Start a DOS command prompt (type `cmd`).
3. From the command prompt type:
telnet www.cityu.edu.hk 80
4. Turn on localecho so you can see what you are typing. **Press Ctrl+] (hold down the Ctrl key and press the right bracket)** to enter the escape mode. Then, type:
set localecho
5. **Press Enter** on a blank line to return to the telnet session. Enter the HTTP request header as follow:
GET / HTTP/1.1
Host: www.cityu.edu.hk
6. **Press Enter twice** to end and send out the request.

What do you see from the response? Repeat this experiment with other websites.

3. With the updated Chrome Dev Tools, you can see the WebSocket traffic coming from and going to your browser without using tools like *Wireshark*. Here are the simple steps to make the invisible visible:

1. Be sure that your Chrome version is 20+.
2. Navigate to the Echo demo, hosted on the websocket.org site.
<http://websocket.org/echo.html>
3. Turn on the Chrome Developer Tools and click **Network** tab.
4. In the Echo demo, click **Connect**.
5. Click the HTTP Request (the red circle) in the right panel.



- Make sure you're on the **Headers** tab. This tab shows the WebSocket handshake where you can find the **Upgrade** header in both request and response.

The screenshot shows the 'Headers' tab in a browser's developer tools. The 'Response Headers' section is expanded, showing the following headers:

- Access-Control-Allow-Credentials: true
- Access-Control-Allow-Headers: x-websocket-protocol
- Access-Control-Allow-Headers: x-websocket-version
- Access-Control-Allow-Headers: x-websocket-extensions
- Access-Control-Allow-Headers: authorization
- Access-Control-Allow-Headers: content-type
- Access-Control-Allow-Origin: http://websocket.org
- Connection: Upgrade**
- Date: Thu, 18 Jan 2018 15:35:29 GMT
- Sec-WebSocket-Accept: fr15va1y+UUzjrLVwcASGi85ADc=
- Server: Kaazing Gateway
- Upgrade: websocket**

The 'Request Headers' section is also expanded, showing the following headers:

- Accept-Encoding: gzip, deflate
- Accept-Language: zh-HK;q=0.9,en-HK;q=0.8,en;q=0.7,zh-TW;q=0.6,en-US;q=0.5,zh-CN;q=0.4
- Cache-Control: no-cache
- Connection: Upgrade**
- Cookie: _ga=GA1.2.263492597.1516289766; _gid=GA1.2.2023530889.1516289766; __zlcmid=kXh3VocKuf6V2M
- DNT: 1
- Host: echo.websocket.org
- Origin: http://websocket.org
- Pragma: no-cache
- Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
- Sec-WebSocket-Key: 1H40QvldNcQEzCaqlZW3+w==
- Sec-WebSocket-Version: 13
- Upgrade: websocket**
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36

- Click the **Send** button in the Echo demo.
- Switch to the **Frames** tab and you should see the messages just being sent as well as its length and time.

Name	× Headers	Frames	Cookies	Timing
☐ ?encoding=text	All	▼ Enter regex, for example: (web)?socket		
Data	Length	Time		
Rock it with HTML5 WebSocket	28	23:44:48.586		
Rock it with HTML5 WebSocket	28	23:44:48.814		

4. Discuss with your classmates and suggest a case for each of the following n-tier application architectures. Briefly explain the reason.

1. 1-tier
2. 2-tier
3. 3-tier
4. 4-tier

- END -

Solution

Q4.

- 1-tier: Simple desktop applications without internet access e.g. painter, calculator.
- 2-tier: Mobile applications with only data store requirement on server. For example, a photo album app that stores and retrieves photos from a server.
- 3-tier: Web applications with separated client, application server and database server. For example, an e-commerce website that processes transactions on the application server.
- 4-tier: Any kinds of applications that requires additional layer of functionality. For example, adding an authentication tier for granting access to user before further processing; or adding an analytic tier for analyzing data from data warehouse.