UNIT 4
HTTP

LH - 2HRS

PREPARED BY: **ER. SHARAT MAHARJAN**

NETWORK PROGRAMMING

PRIME COLLEGE, NAYABAZAAR

CONTENTS (LH - 2HRS)

4.1 The protocol: Keep-Alive

4.2 HTTP Methods

4.3 The Request Body

4.4 Cookies: CookieManager and CookieStore

4.1 The protocol: Keep-Alive

A typical client request :

GET /index.html HTTP/1.1

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:20.0) Gecko/20100101 Firefox/20.0

Host: en.wikipedia.org

Connection: keep-alive

Accept-Language: en-US,en;

Accept-Encoding: gzip, deflate

Accept: text/html

A typical successful response :

HTTP/1.1 **200 OK**

Date: Sun, 21 Apr 2013 15:12:46 GMT

Server: Apache

Connection: close

Content-Type: text/html

Content-length: 115

A response code :

✓100 to 199 always indicates an informational response

✓200 to 299 always indicates success

✓300 to 399 always indicates redirection

✓400 to 499 always indicates a client error

✓500 to 599 indicates a server error

HTTP 1.0 opens a **new connection for each request**.

- The time taken to open and close all the connections in a typical web session can outweigh the time taken to transmit the data, especially for sessions with many small documents.

In HTTP 1.1 and later, the **server doesn't have to close the socket after it sends its response**.

A client indicates that it's willing to **reuse a socket** by including a Connection field in the HTTP request header with the value Keep-Alive:

Connection: Keep-Alive

4.2 HTTP Methods

- **Communication** with an HTTP server follows a **request-response pattern**.
- Each **HTTP request** has **two or three parts**:

1. **Start line** containing the **HTTP method and a path to the resource**.
2. **Header of name-value fields** that provide meta-information.
3. **Request body** containing a representation of a resource (**POST and PUT only**)

- There are **four main HTTP methods**, operations that can be performed:

1. **GET**
2. **POST**
3. **PUT**
4. **DELETE**

4.3 The Request Body

- The **GET method retrieves a representation of a resource** identified by a URL.
- **POST and PUT are more complex**. In these cases, the **client supplies the representation of the resource**, in addition to the path and the query string.
- The **representation of the resource** is sent in the **body of the request**, after the header.
- That is, it sends these four items in order:

1. A **starter line** including the **method**, **path and query string**, and **HTTP version**
2. An **HTTP header**
3. A **blank line**
4. The **body**

- **For example**, this **POST request** sends **form data to a server**:

POST /cgi-bin/register.pl HTTP 1.0

Date: Sun, 27 Apr 2013 12:32:36

Host: www.cafeaulait.org

Content-type: application/x-www-form-urlencoded

Content-length: 54

username=Elliotte+Harold&email=elharo%40ibiblio.org

- However, **the HTTP header should include two fields** that specify the nature of the body:

✓A **Content-length field** that specifies how many bytes are in the body.

✓A **Content-type field** that specifies the MIME media type of the bytes

4.4 Cookies: CookieManager and CookieStore

- Many websites use small strings of text known as **cookies to store persistent client-side state between connections.**
- Cookies are **passed from server to client and back again in the HTTP headers of requests and responses.**
- Cookies are limited to **nonwhitespace ASCII text**, and **may not contain commas or semicolons**.
- To set a cookie in a browser, the **server includes a Set-Cookie header line** in the HTTP header.
- **For example**, this HTTP header **sets the cookie "cart" to the value "ATVPDKIKX0DER":**

HTTP/1.1 200 OK

Content-type: text/html

Set-Cookie: cart=ATVPDKIKX0DER

- If a browser makes a **second request to the same server**, it will **send the cookie back in a Cookie line in the HTTP request header** like so:

GET /index.html HTTP/1.1

Host: www.example.org

Cookie: cart=ATVPDKIKX0DER

Accept: text/html

- This enables the server to **track individual users and sessions across multiple HTTP connections**. Servers can set **more than one cookie**.
- **For example**, a request made to Amazon fed browser five cookies:

Set-Cookie:skin=noskin

Set-Cookie:ubid-main=176-5578236-9590213

Set-Cookie:session-token=Zg6afPNqbaMv2WmYFOv57zCU1O6Ktr

Set-Cookie:session-id-time=2082787201l

Set-Cookie:session-id=187-4969589-3049309

- In addition to a simple name=value pair, cookies can have **several attributes that control their scope including:**

✓expiration date,

✓path,

✓domain,

✓port,

✓version, and

✓security options

- Java 6 includes java.net.CookieManager subclass of CookieHandler can be use.
- Before Java will store and return cookies, you need to enable it:

```
CookieManager manager = new CookieManager();
CookieHandler.setDefault(manager);
```

- Three policies are predefined:

✓CookiePolicy.ACCEPT_ALL: All cookies allowed

✓CookiePolicy.ACCEPT_NONE: No cookies allowed

✓CookiePolicy.ACCEPT_ORIGINAL_SERVER: Only first party cookies allowed

- For example, this code fragment tells Java to block third-party cookies but accept first party cookies:

```
CookieManager manager = new CookieManager();
manager.setCookiePolicy(CookiePolicy.ACCEPT_ORIGINAL_SERVER);
CookieHandler.setDefault(manager);
```

- We can retrieve the store in which the CookieManager saves its cookies with the getCookieStore() method:

CookieStore store = manager.getCookieStore();

- The CookieStore class allows you to add, remove, and list cookies so you can control the cookies that are sent outside the normal flow of HTTP requests and responses.

THANK YOU FOR YOUR ATTENTION