IEMS5722
Mobile Network Programming and Distributed Server Architecture

Lecture 4

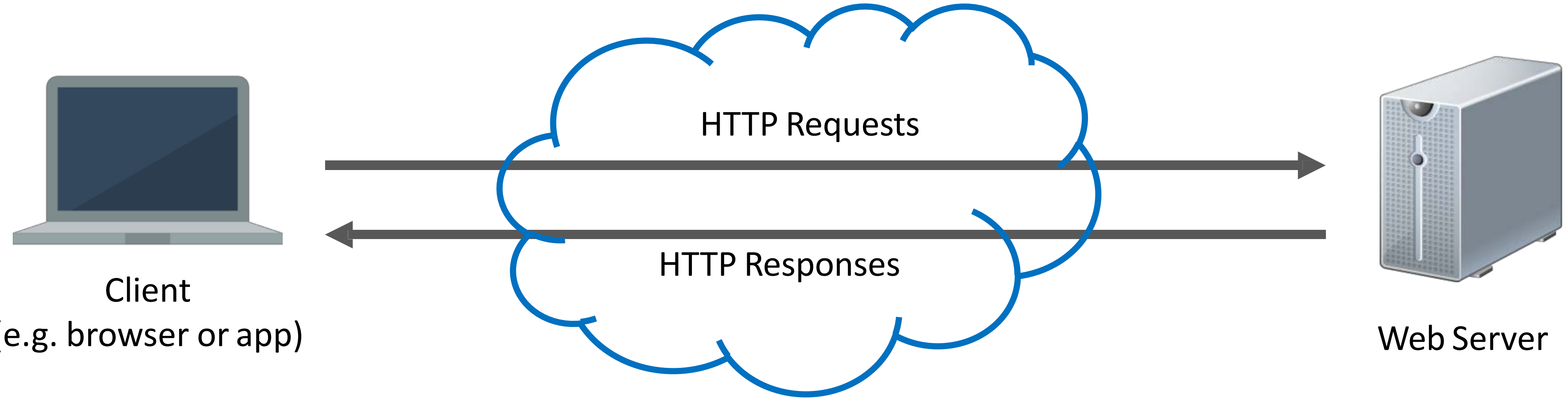
HTTP Networking in Android

Online Applications

- Examples?
  - CUHK Website (https://www.cuhk.edu.hk/)
  - Facebook Website (https://www.facebook.com/)
  - HKO’s RSS Feed (https://rss.weather.gov.hk/)
  - Instagram App
  - Twitter REST API
  - …

Online Applications

- All the above services make use of the HTTP protocol

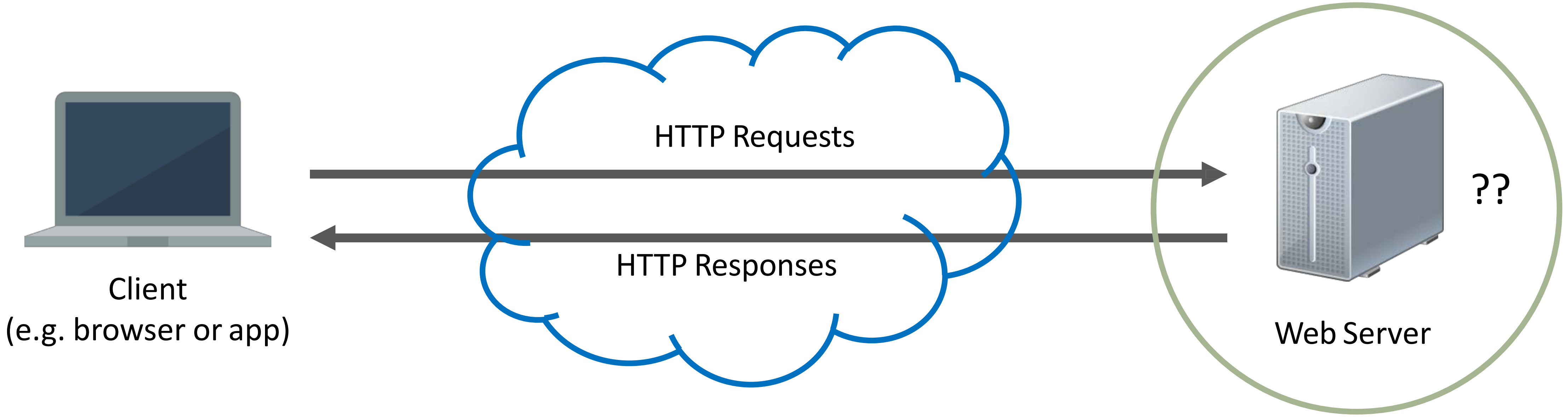


- Let's study more about HTTP first!

Hypertext Transfer Protocol HTTP

HTTP

- What is HTTP (Hypertext Transfer Protocol)?
- What happen between a request is made and a response is received?



HTTP

- Tim Berners-Lee, credited as the inventor of the World Wide Web, created the original HTTP and HTML in 1990 at CERN
- For combining the **Internet** and **hypertext**

Tim Berners-Lee

The first Web server

HTTP

- An application protocol for transferring hypertext and other file formats over the Internet
- Current widely used version: HTTP/1.1 (standardized in 1997)
- HTTP/2 specification was published as RFC 7540 in May 2015
- Client (e.g. Web browser) sends an HTTP **request** to a **URL**
- Server prepares and returns the requested **resources**

HTTP Requests

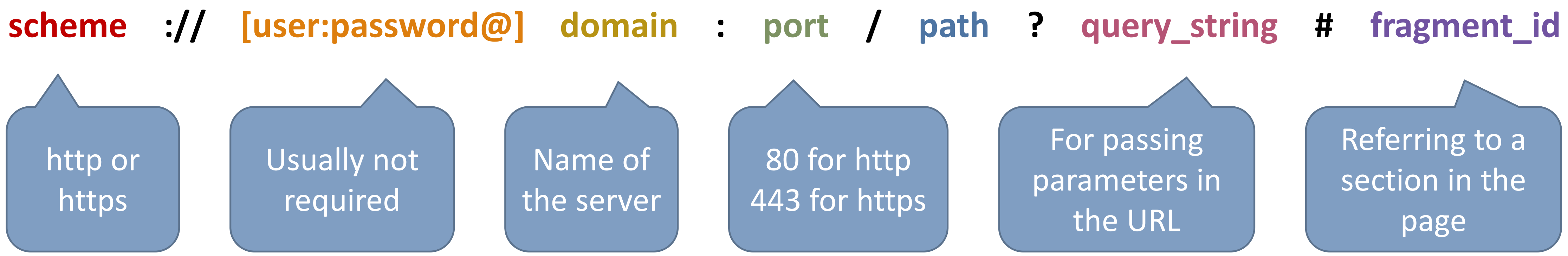
- An HTTP request has the following components
  - **URL** – the unique identifier of the online resource
  - **Method/Verb** – the action of the request (e.g. GET something?)
  - **HTTP Version** – the version of the protocol you are using
  - **Headers** – the metadata of the request
  - **Body** – Data to be sent to the server

HTTP Response

- An HTTP response has the following components
  - **Status Code** – indicate whether the request is successful
  - **HTTP Version** – the version of the protocol you are using
  - **Headers** – metadata of the response
  - **Body** – data of the resource requested

URL

- Uniform Resource Locator (URL)
  - A specific type of URI (Uniform resource identifier)
  - It implies the means to access a resource
  - Syntax of a URL:


scheme :// [user:password@] domain : port / path ? query_string # fragment_id
http or https
Usually not required
Name of the server
80 for http 443 for https
For passing parameters in the URL
Referring to a section in the page

URL

- Examples:
  - CUHK Homepage
    http://www.cuhk.edu.hk/english/index.html
  - YouTube Video
    https://www.youtube.com/watch?v=21KLWN29RiA
  - Hong Kong Observatory Radar Image
    https://www.hko.gov.hk/en/wxinfo/radars/radar.htm?pv_mode=playback
  - Instagram API
    https://api.instagram.com/v1/self/media/recent?access_token=ACCESS_TOKEN

HTTP Request Methods

- Indicate the desired action to be performed on the resource identified by the URL
  - **GET** – retrieves data from the server
  - **HEAD** – asks for a response same as GET, but without the body
  - **POST** – asks the server to accept data enclosed in the request and apply it to the resource
  - **PUT** – asks the server to store the data under the supplied URL
  - Other methods: DELETE, TRACE, OPTIONS, CONNECT, PATCH

HTTP Request Methods

- An example of **GET**:
  - https://www.youtube.com/watch?v=21KLWN29RiA
    - Retrieve a YouTube video page providing the value of the parameter v
    - It has no effect on the resource to be retrieved, it simply retrieves a copy of the resource

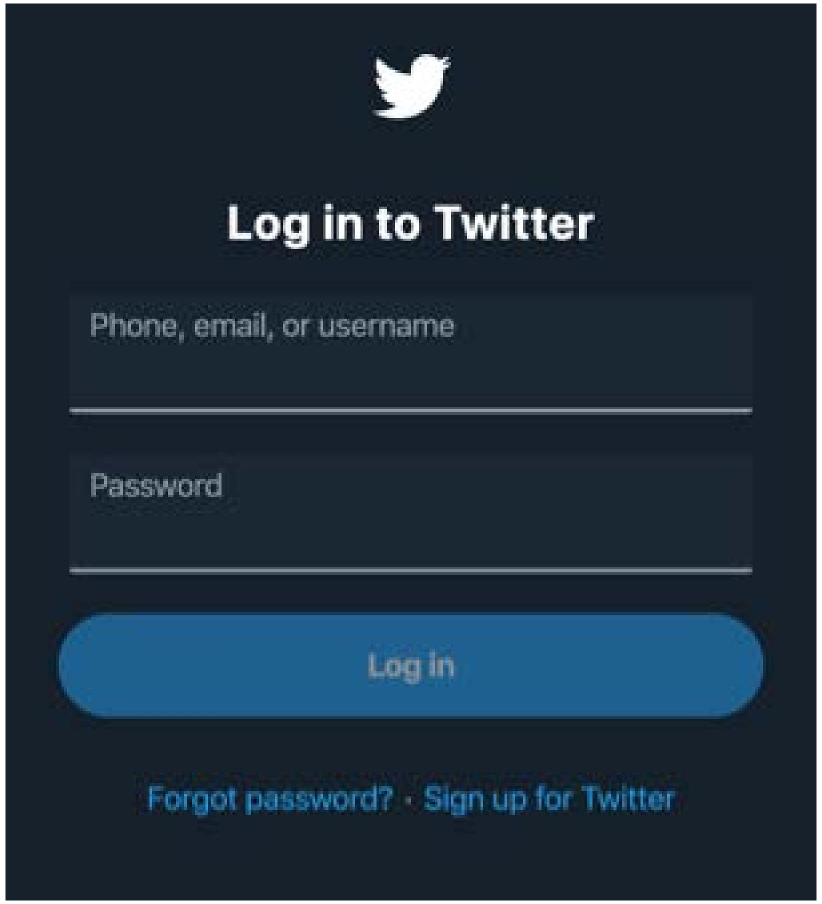
  - "v=21KLWN29RiA" is the **query string**

HTTP Request Methods

- **Query String**
  - Each parameter and its value are specified by **name**=**value**
  - Parameters are separated by ampersand “**&**”
  - The maximum amount of information that can be passed to the server using the query string depends on the maximum length allowed for an URL (The limits of different browsers are different, usually at about 64k characters)
  - NOT for passing sensitive data (e.g. password)

HTTP Request Methods

- An example of POST:
- https://twitter.com/login
- After filling in the username and password and clicking on the "Log in" button, the data will be sent to Twitter's server using the "**POST**" method
- Usually used for submitting a form (e.g. online forms, leaving comments, etc.)


Log in to Twitter
Phone, email, or username
Password
Log in
Forgot password? · Sign up for Twitter

HTTP Request Methods

- Recall that HTTP is a text protocol (i.e. everything sent using HTTP are assumed to be characters)
- If you want to send files (binary data), you need to encode the binary data first before sending
- In an HTML form, set **enctype="multipart/form-data"** (see next slide)

HTTP Request Methods

```
<form method="post" enctype="multipart/form-data">
    <input type="text" name="name">
    <input type="file" name="file">
    <input type="submit" value="Send!">
</form>
```

- Setting **enctype="multipart/form-data"** tells the server that the data are split into multiple parts, one for each file, plus one for the textual data in the form body.

Reference: https://developer.mozilla.org/en-US/docs/Learn/Forms/Sending_and_retrieving_form_data

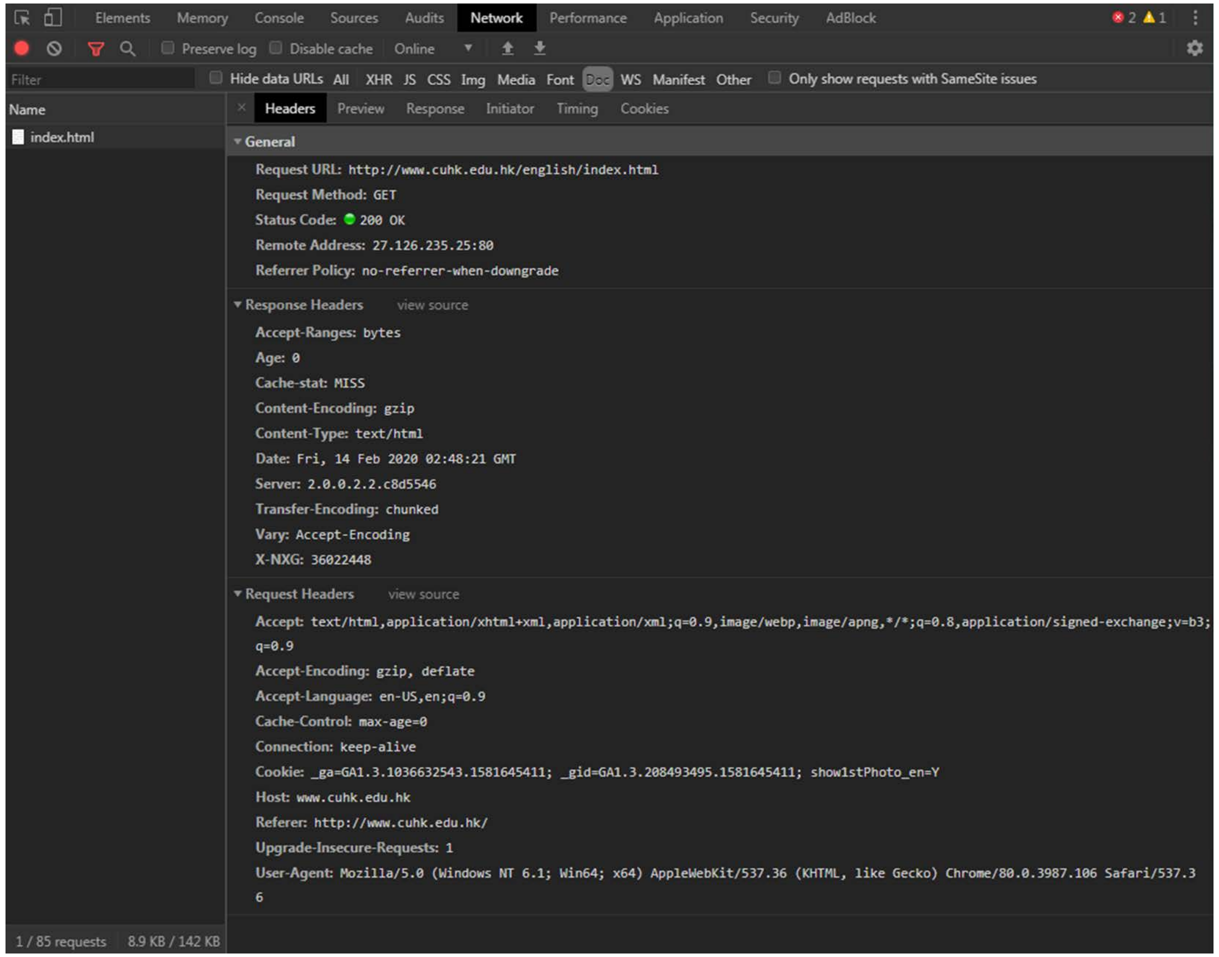
HTTP Headers

- Headers contain metadata about the request/response, such as:
  - Identity of the client
  - Type of the content (e.g. plain text, HTML, CSS, image)
  - Encoding of the content (e.g. ASCII, utf-8)
  - Expiry date/time of the content
  - Cookies
  - …

- For a list of HTTP request and response header fields, see: https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

Inspecting HTTP Requests and Responses

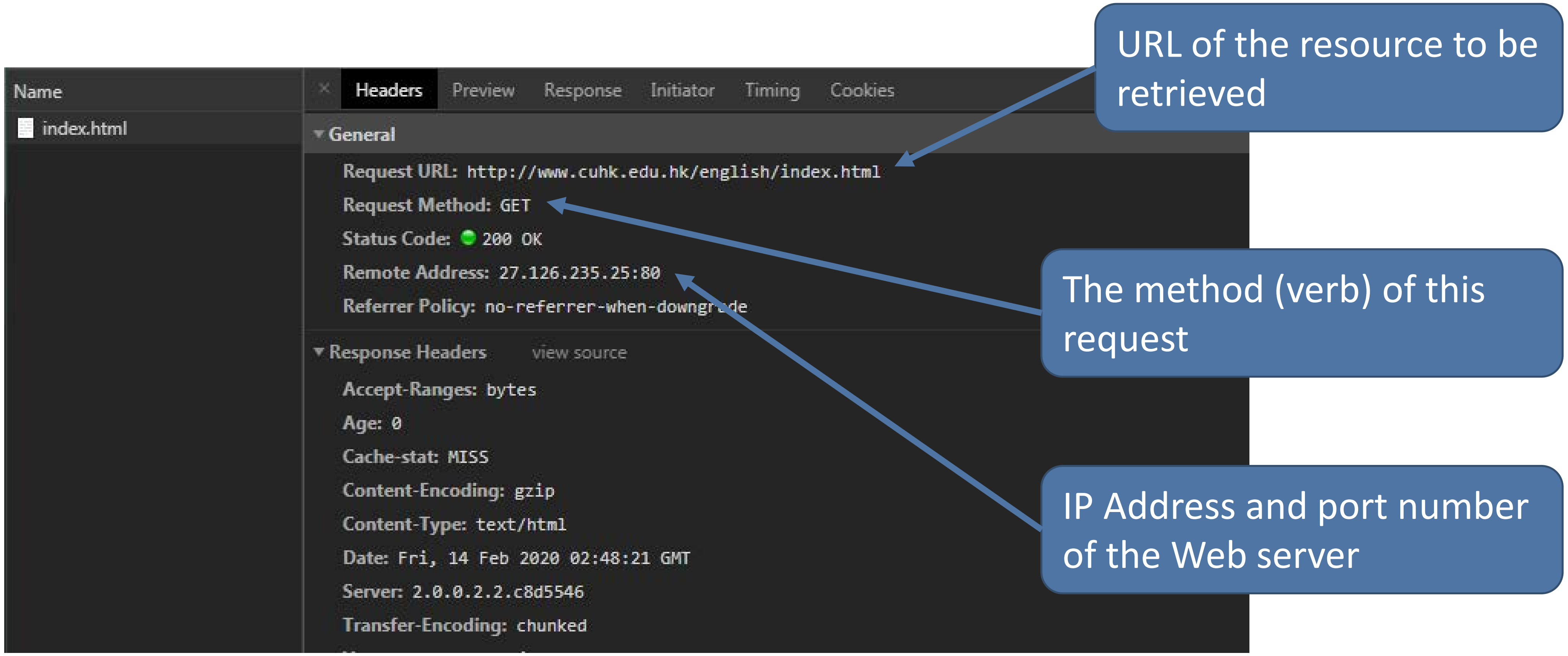
- Use the developer's tools in Firefox or Chrome:



Elements Memory Console Sources Audits Network Performance Application Security AdBlock
Preserve log Disable cache Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other Only show requests with SameSite issues
Name
index.html
Headers Preview Response Initiator Timing Cookies
1 / 85 requests 8.9 KB / 142 KB

```
▼ General
  Request URL: http://www.cuhk.edu.hk/english/index.html
  Request Method: GET
  Status Code: ● 200 OK
  Remote Address: 27.126.235.25:80
  Referrer Policy: no-referrer-when-downgrade

▼ Response Headers      view source
  Accept-Ranges: bytes
  Age: 0
  Cache-stat: MISS
  Content-Encoding: gzip
  Content-Type: text/html
  Date: Fri, 14 Feb 2020 02:48:21 GMT
  Server: 2.0.0.2.2.c8d5546
  Transfer-Encoding: chunked
  Vary: Accept-Encoding
  X-NXG: 36022448

▼ Request Headers      view source
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
  q=0.9
  Accept-Encoding: gzip, deflate
  Accept-Language: en-US,en;q=0.9
  Cache-Control: max-age=0
  Connection: keep-alive
  Cookie: _ga=GA1.3.1036632543.1581645411; _gid=GA1.3.208493495.1581645411; show1stPhoto_en=Y
  Host: www.cuhk.edu.hk
  Referer: http://www.cuhk.edu.hk/
  Upgrade-Insecure-Requests: 1
  User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.106 Safari/537.3
  6
```

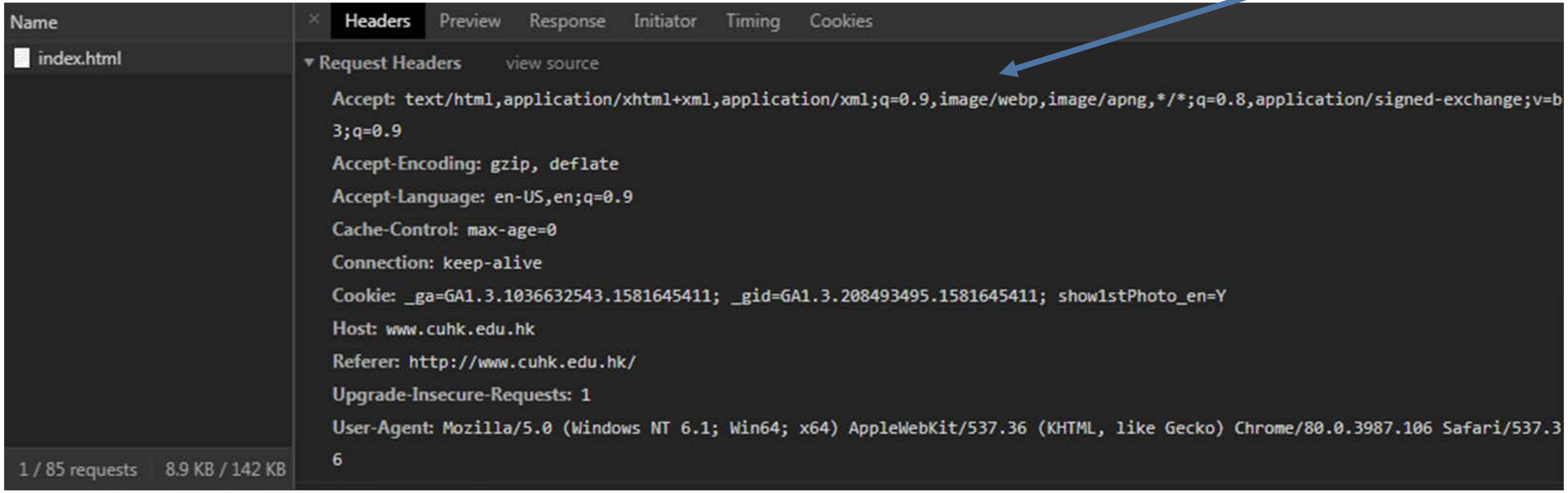
Example: CUHK Homepage


Name
index.html
Headers
Preview
Response
Initiator
Timing
Cookies
URL of the resource to be retrieved
The method (verb) of this request
IP Address and port number of the Web server

```
▼ General
    Request URL: http://www.cuhk.edu.hk/english/index.html
    Request Method: GET
    Status Code: ● 200 OK
    Remote Address: 27.126.235.25:80
    Referrer Policy: no-referrer-when-downgrade
▼ Response Headers      view source
    Accept-Ranges: bytes
    Age: 0
    Cache-stat: MISS
    Content-Encoding: gzip
    Content-Type: text/html
    Date: Fri, 14 Feb 2020 02:48:21 GMT
    Server: 2.0.0.2.2.c8d5546
    Transfer-Encoding: chunked
```

Example: CUHK Homepage

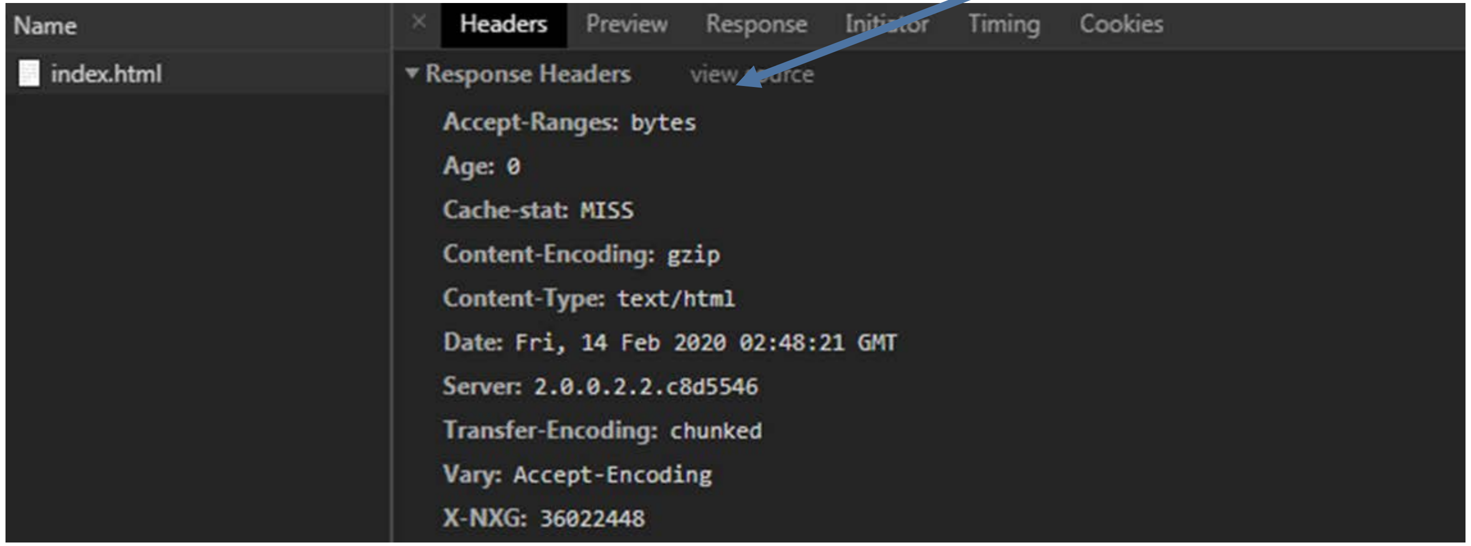
Headers of the request



```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b
3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
Cookie: _ga=GA1.3.1036632543.1581645411; _gid=GA1.3.208493495.1581645411; show1stPhoto_en=Y
Host: www.cuhk.edu.hk
Referer: http://www.cuhk.edu.hk/
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.106 Safari/537.3
6
```

Example: CUHK Homepage

Headers of the response

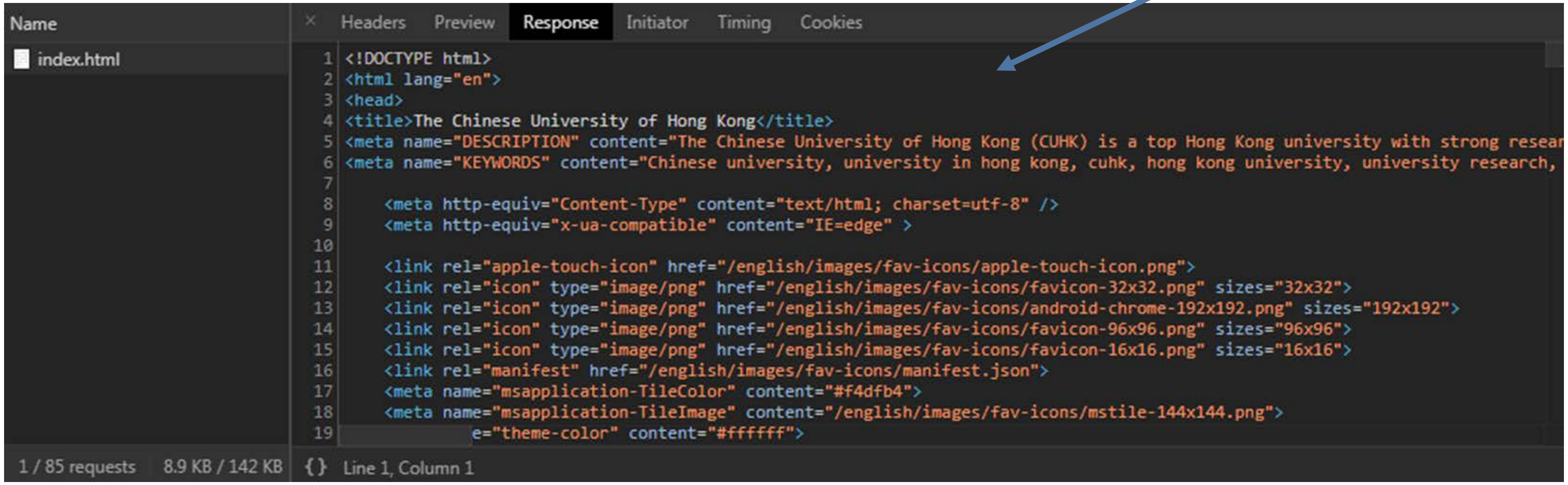


▾ Response Headers

```
Accept-Ranges: bytes
Age: 0
Cache-stat: MISS
Content-Encoding: gzip
Content-Type: text/html
Date: Fri, 14 Feb 2020 02:48:21 GMT
Server: 2.0.0.2.2.c8d5546
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-NXG: 36022448
```

Example: CUHK Homepage

Content of the response



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>The Chinese University of Hong Kong</title>
<meta name="DESCRIPTION" content="The Chinese University of Hong Kong (CUHK) is a top Hong Kong university with strong resear
<meta name="KEYWORDS" content="Chinese university, university in hong kong, cuhk, hong kong university, university research,

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta http-equiv="x-ua-compatible" content="IE=edge" >

    <link rel="apple-touch-icon" href="/english/images/fav-icons/apple-touch-icon.png">
    <link rel="icon" type="image/png" href="/english/images/fav-icons/favicon-32x32.png" sizes="32x32">
    <link rel="icon" type="image/png" href="/english/images/fav-icons/android-chrome-192x192.png" sizes="192x192">
    <link rel="icon" type="image/png" href="/english/images/fav-icons/favicon-96x96.png" sizes="96x96">
    <link rel="icon" type="image/png" href="/english/images/fav-icons/favicon-16x16.png" sizes="16x16">
    <link rel="manifest" href="/english/images/fav-icons/manifest.json">
    <meta name="msapplication-TileColor" content="#f4dfb4">
    <meta name="msapplication-TileImage" content="/english/images/fav-icons/mstile-144x144.png">
             e="theme-color" content="#ffffff">
```

More on HTTP Headers

- HTTP headers are sets of key-value pairs (field names and values)

- Some of the **request header** “keys”:
  - Accept: the preferred format of the resource
    (e.g. text/html, application/json, application/xml)
  - Accept-Language: the preferred language of the resource
    (e.g. zh-TW, zh-CN, en-US)
  - User-Agent: the type of browser or device
    (e.g. indicate whether the client is on a PC or on a mobile)

More on HTTP Headers

- Some of the **response header** "keys":
  - Content-Length: length of the content of the resource
  - Content-Type: format of the resource (e.g. text/html)
  - Last-Modified: the time when the resource was last changed
  - Server: The name of the Web server serving the resource

- For a comprehensive list of header fields: https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

HTTP Status Code

- HTTP status code is included in a HTTP response to indicate the outcome of an HTTP request

- The different categories of HTTP status codes:
  - 1XX: Informational
  - 2XX: Successful
  - 3XX: Redirection
  - 4XX: Client-side error
  - 5XX: Server-side error

HTTP Status Code

- Examples of HTTP status codes
  - **200: OK**
    Everything is OK, results should be in the response
  - **301: Moved Permanently**
    The client should send request from the URL provided instead
  - **403: Forbidden**
    The client is not authorized to access the resource
  - **404: Not Found**
    The resource cannot be found
  - **500: Internal Server Error**
    Some problem with your server application

References

- In Introduction to HTTP Basics https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

Stateless-ness

- HTTP is a **stateless** protocol
  - The server does not retain information about clients between requests
  - Each request is considered independent
  - No session information stored on the server-side

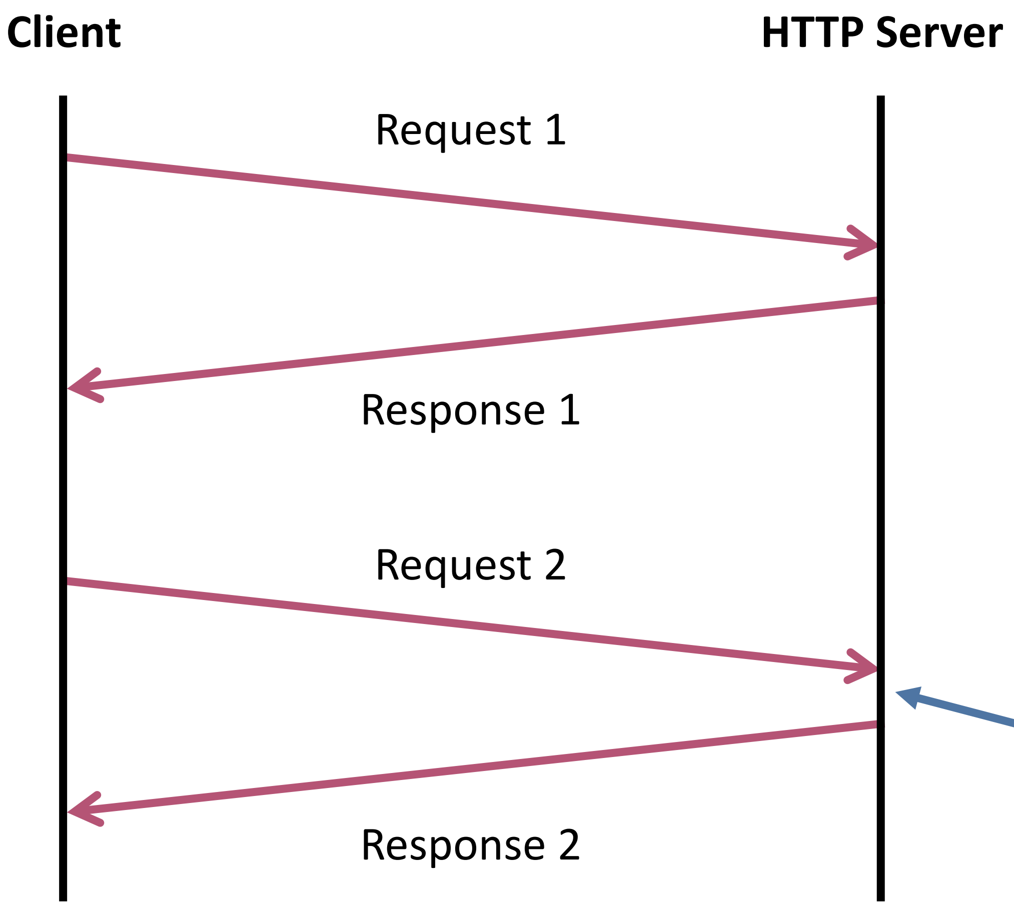
- (See illustration on the next slide)

Stateless-ness

In order to let the server know that the client has done something before, the client has to include some information in the request (header or content)

Example:

- The client has already logged in
- The client has filled in a form but some fields are missing



At this point, the server knows nothing about the content and outcome of Request 1.

Using HTTP in Android

Data Communication using HTTP

- How can we perform HTTP requests to a Web server in Android?
  - First of all, you need to ask for permission in the AndroidManifest.xml file

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- Use the **HttpURLConnection** HTTP client to perform HTTP requests

* Note: some online resources may mention the use the DefaultHTTPClient class, however that class should not be used as it is deprecated.

HttpURLConnection

- **HttpURLConnection** can be used to perform both GET and POST actions
  - Check the following link for usage examples https://developer.android.com/training/basics/network-ops/connecting.html
  - Data is returned in the form of **InputStream**
  - Depending on the data type of the data (e.g. image, text, file, etc.), you need to **decode** the data into appropriate format

HttpURLConnection

- Example:

  Performing a GET request to http://www.cuhk.edu.hk

```
InputStream is = null;

try {
    URL url = new URL("http://www.cuhk.edu.hk");
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setReadTimeout(10000); // 10,000 milliseconds
    conn.setConnectTimeout(15000); // 15,000 milliseconds
    conn.setRequestMethod("GET"); // Use the GET method
    conn.setDoInput(true);
    // Start the query
    conn.connect();
    int response = conn.getResponseCode(); // This will be 200 if successful
    is = conn.getInputStream();

    // Convert the InputStream into a string
    String results = "";
    String line;
    BufferedReader br = new BufferedReader(new InputStreamReader(is));
    while (line = br.readLine() != null) {
        results += line;
    }
} finally {
    if (is != null) {
        is.close(); // Close the InputStream when done
    }
}
```

HttpURLConnection

- Example:

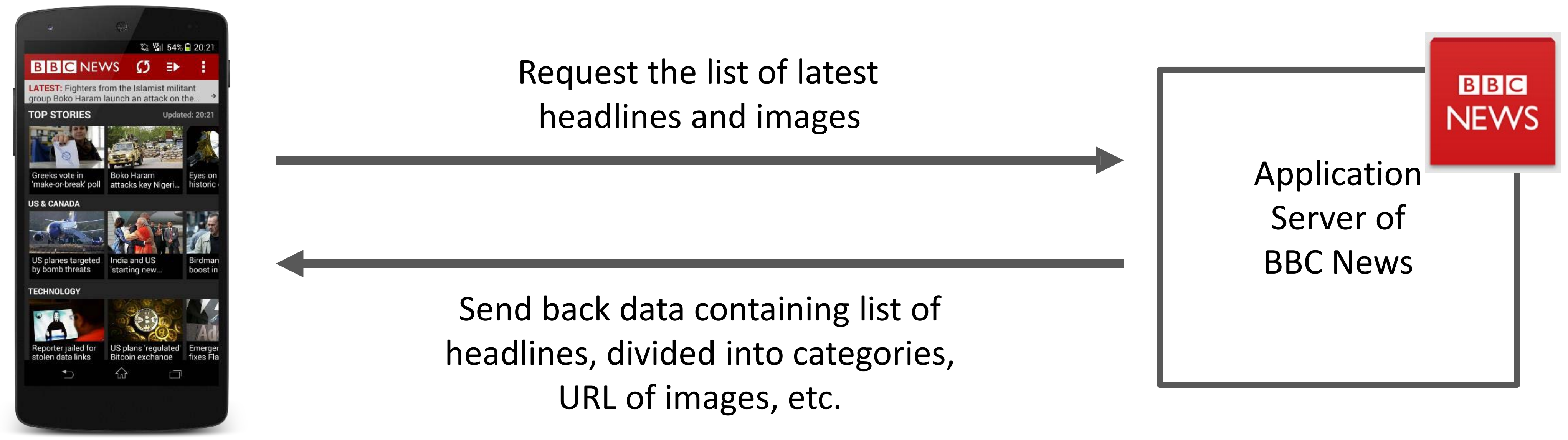
  Performing a POST request to http://www.example.com/submit_form

```
URL url = new URL("http://www.example.com/submit_form");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setReadTimeout(15000);
conn.setConnectTimeout(15000);
conn.setRequestMethod("POST");
conn.setDoInput(true);
conn.setDoOutput(true);

OutputStream os = conn.getOutputStream();
BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, "UTF-8"));
Uri.Builder builder = new Uri.Builder();

// Build the parameters using ArrayList objects para_names and para_values
for (int i = 0; i < para_names.size(); i++) {
    builder.appendQueryParameter(para_names.get(i), para_values.get(i));
}
String query = builder.build().getEncodedQuery();

writer.write(query);
writer.flush();
writer.close();
os.close();

int responseCode = conn.getResponseCode();
if (responseCode == HttpURLConnection.HTTP_OK) {
    // Process the response
}
```

Exchanging Data Between Server & Client

Exchange Data

- In app programming, very often we are not requesting Web pages. Instead, we request data from the server, such as:
  - List of latest news (news app)
  - History of conversation (instant messaging app)



BBC NEWS
LATEST: Fighters from the Islamist militant group Boko Haram launch an attack on the...
TOP STORIES
Updated: 20:21
Greeks vote in 'make-or-break' poll
Boko Haram attacks key Nigeri...
US & CANADA
US planes targeted by bomb threats
India and US 'starting new...
TECHNOLOGY
Reporter jailed for stolen data links
US plans 'regulated' Bitcoin exchange
Request the list of latest headlines and images
Send back data containing list of headlines, divided into categories, URL of images, etc.
Application Server of BBC News
BBC NEWS

Exchange Data

- In order to exchange structured data, we need to have a common data format Common data exchange formats include:
  1. **XML (Extensible Markup Language)**
     - Using different tags (e.g. <title></title>) to give meanings to the data
     - May result in a significant increase in the length of the data
  2. **JSON (JavaScript Object Notation)**
     - JavaScript objects encoded as strings, can handle several data types such as strings, numbers, Booleans and arrays.
     - More compact compared to XML, still easy to read by human

JSON

- Below is an example of data coded in JSON format

```
{
    "status": "100",
    "message": "Request processed without error.",
    "data": [
        {
            "title": "News title 001",
            "content": "..."
        },
        {
            "title": "News title 002",
            "content": "...”
        },
        {
            "title": "News title 003",
            "content": "...”
        }
    ]
}
```

Objects are indicated by {…}

Arrays are indicated by […]

All values should be enclosed by quotation marks "…"

Parsing JSON Data

- If you have a string containing encoded JSON data, you can extract data by using the following method:

```
JSONObject json = new JSONObject(json_string);

int status = json.getInt("status");
String message = json.getString("message");

JSONArray array = json.getJSONArray("data");
for (int i = 0; i < array.length(); i++) {
    String title = array.getJSONObject(i).getString("title");
    ...
    ...
}
```

- **Note**: There is a possibility that the string is not well formatted, you have to put these codes inside a try/catch block to catch **JSONExceptions**

Creating JSON Objects

- You can also create a JSONObject and populate it with data

```
JSONObject json = new JSONObject();
json.put("title", "...");
json.put("content", "...");

JSONObject json2 = new JSONObject();
json2.put("x", "...");
json2.put("y", "...");

json.put("data", json2);

String json_string = json.toString();
```

- **Note**: There is a possibility that the string is not well formatted, you have to put these codes inside a try/catch block to catch **JSONExceptions**

JSON for Data Communication

- JSON has been used in many APIs for returning structured data.
  - For example, Current Weather Report API from Hong Kong Observatory serves JSON data
  - https://data.weather.gov.hk/weatherAPI/opendata/weather.php?dataType=rhrread&lang=en

```
{
    "rainfall":
        {
            "data":[
                {
                    "unit":"mm",
                    "place":"Central &amp; Western District",
                    "max":2,
                    "min":0,
                    "main":"FALSE”
                },
...
```

More on Multi-threading in Android

Performing HTTP Requests

- Very often, you need to perform network operations in the background, and then display the data fetched from the network to the users
- The **two rules** again:
  - You should not block the UI thread
  - You should not manipulate UI components from other threads

Performing HTTP Requests

- One option is to use Runnable and the View.post() function:

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            text = fetchDatafromNetwork();
            ...
            ...
            textView.post(new Runnable() {
                public void run() {
                    textView.setText(text);
                }
            });
        }
    }).start();
}
```

Performing HTTP Requests

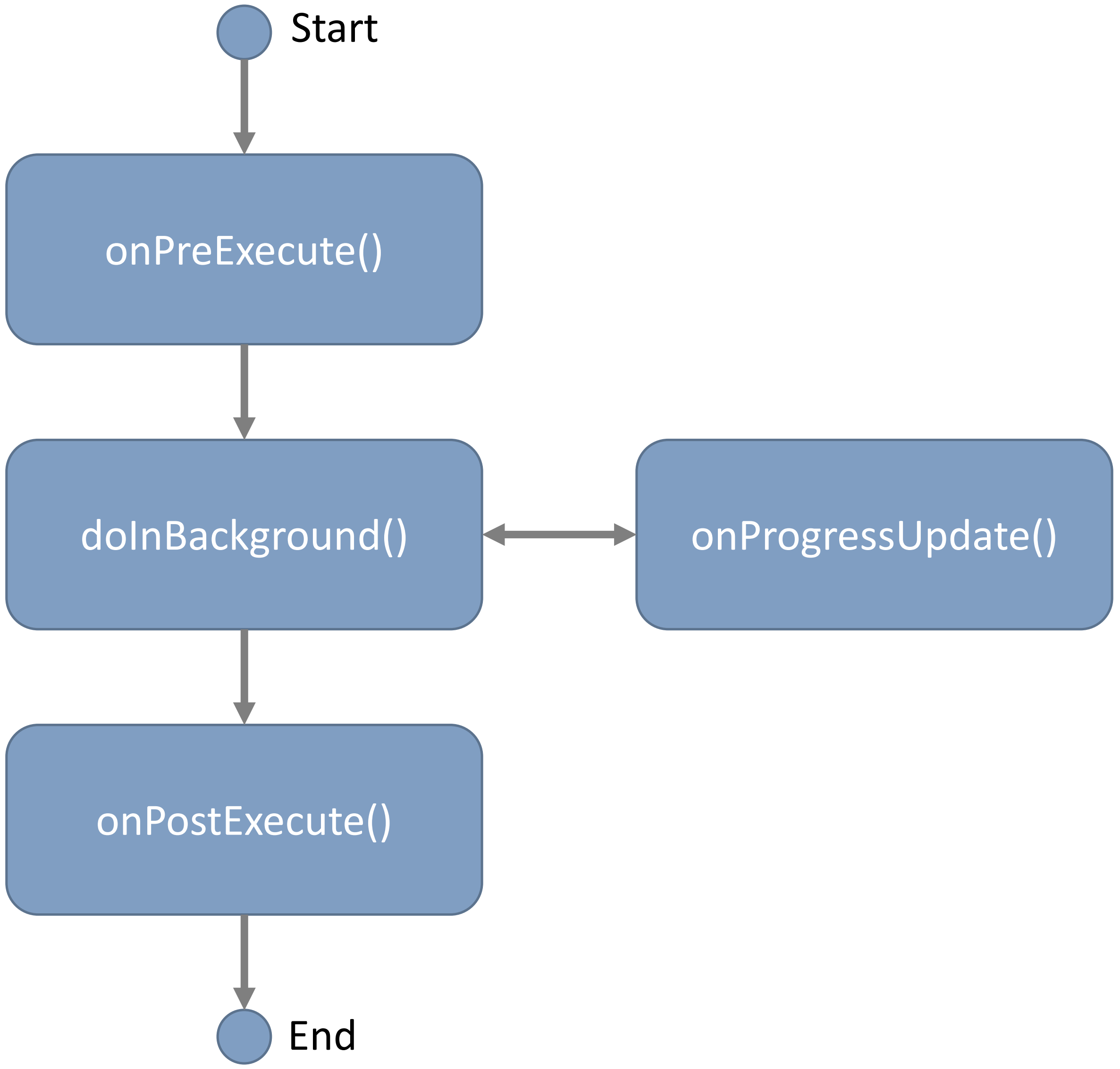
- The **down-side** of using Runnable + View.post():
  - Codes become difficult to manage
  - Need to design parameter passing, or using a lot of global variables
  - Not suitable if you need to update a lot of UI components after the network operation

AsyncTask

- **AsyncTask** provides a proper and easy-to-use method to perform background operations and manipulate UI components, without worrying about creating threads.
  - It allows you to perform **asynchronous** tasks in the background
  - It is suitable for **short operations** (e.g. a few seconds)
  - It **must be sub-classed** to be used (extends AsyncTask)

AsyncTask

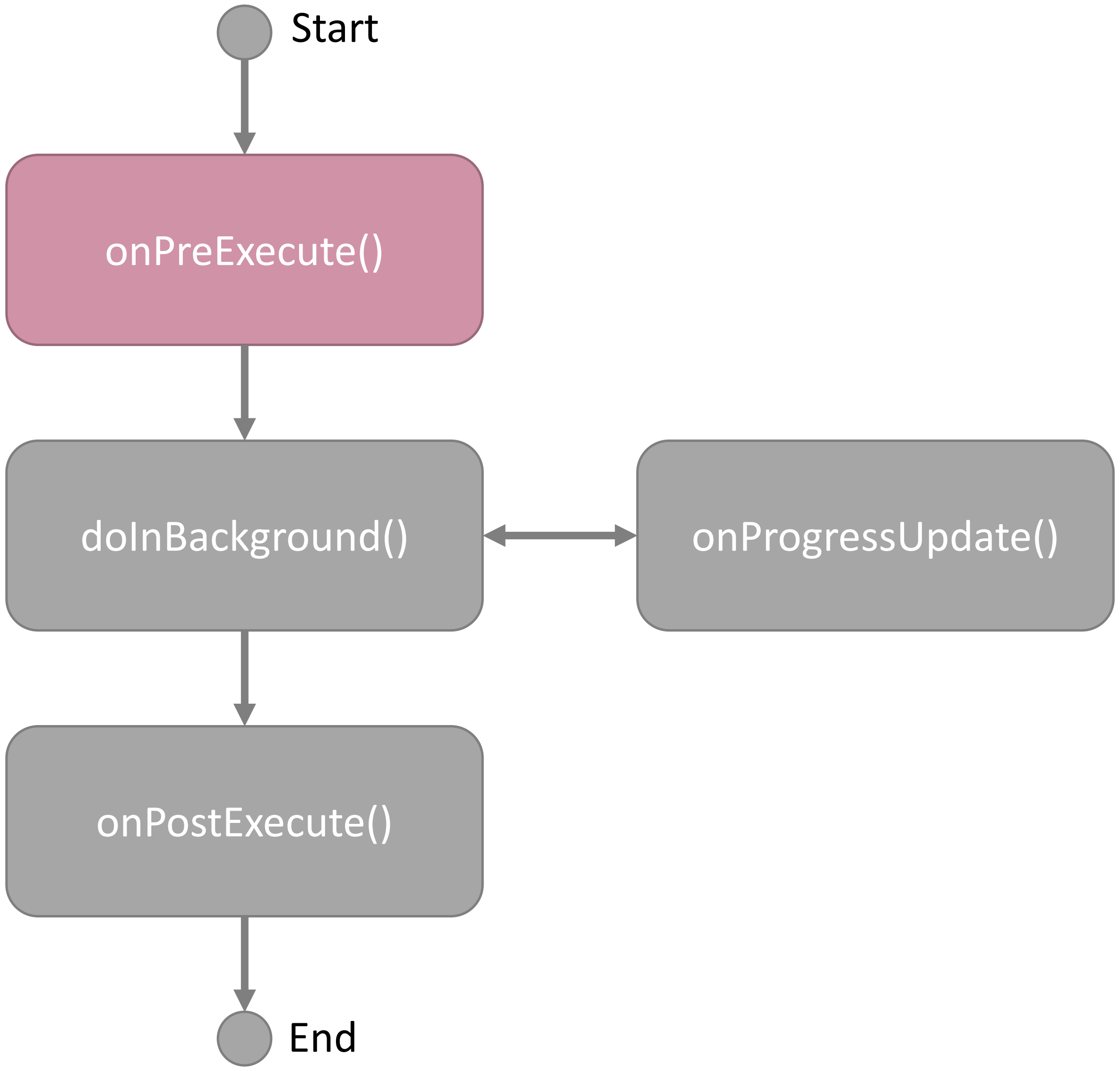
- 4 Steps in an AsycnTask
  - onPreExecute()
  - doInBackground()
  - onProgressUpdate()
  - onPostExecute()



AsyncTask

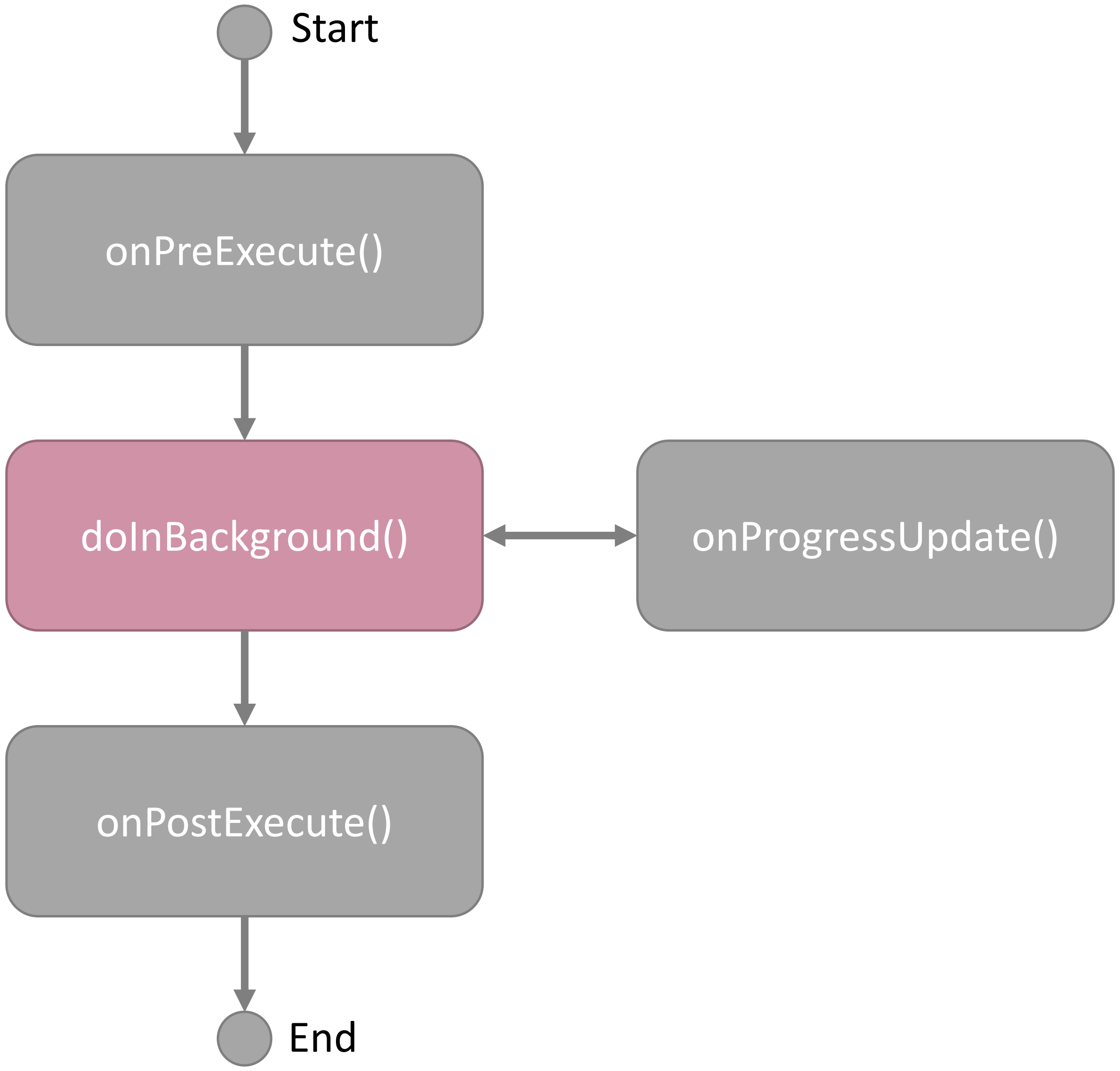
- onPreExecute(): Do preparations before the operation is performed, for example:
  - Start displaying a progress dialog (e.g., now loading…, please wait…)
  - Initialize variables
  - Etc.

- **NOTE**:
  This function runs on the **UI thread**



AsyncTask

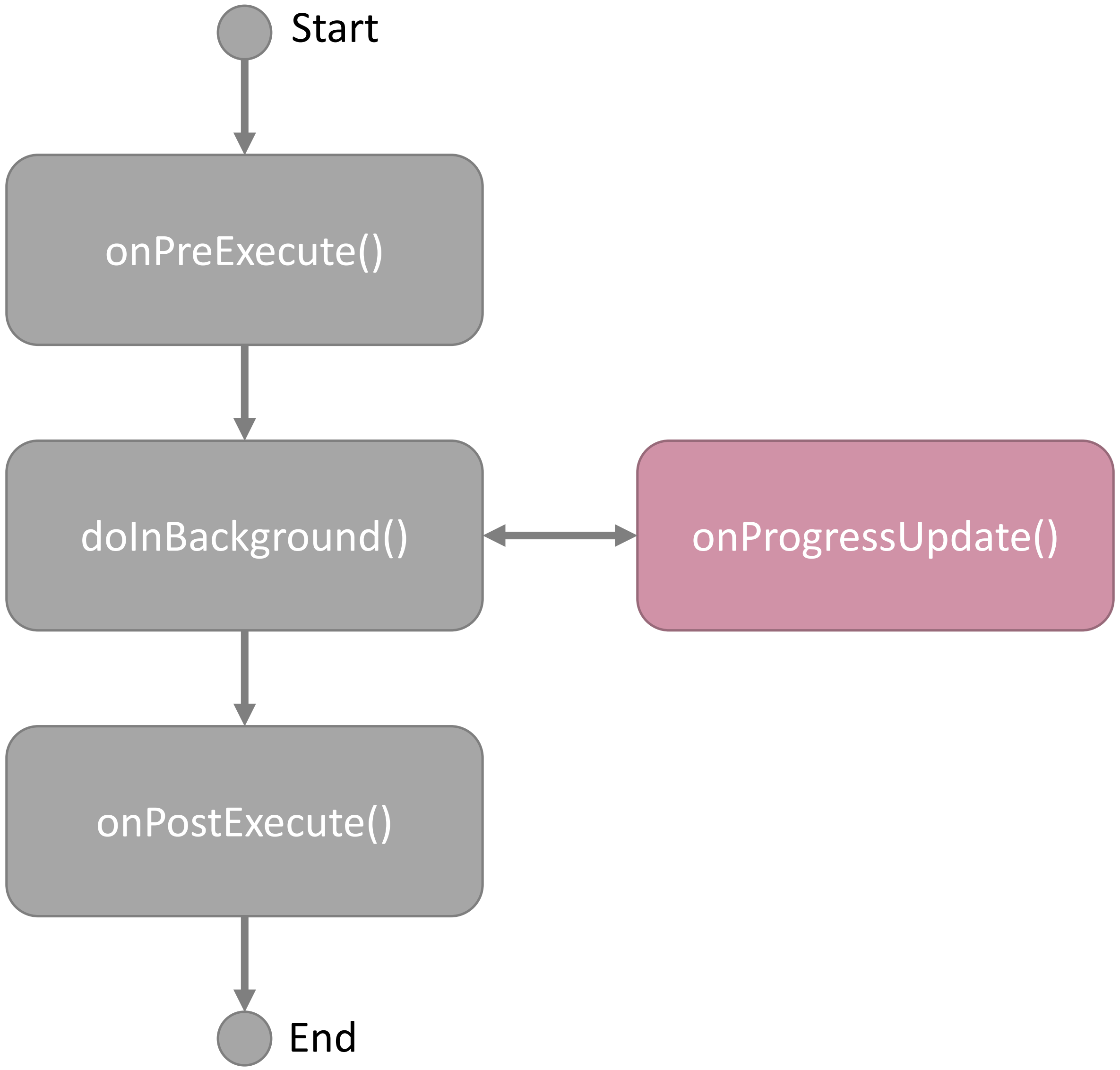
- doInBackground(): Perform the background operation that takes some time, for example:
  - Connecting to a server to fetch data
  - Performing some heavy calculations/operations
- **NOTE**:
  This function runs on a **new thread**



AsyncTask

- onProgressUpdate(): When you call publishProgress() in doInBackground(), this function will be called, here you may:
  - Update the percentage in the progress dialog
  - Show intermediate result to the user

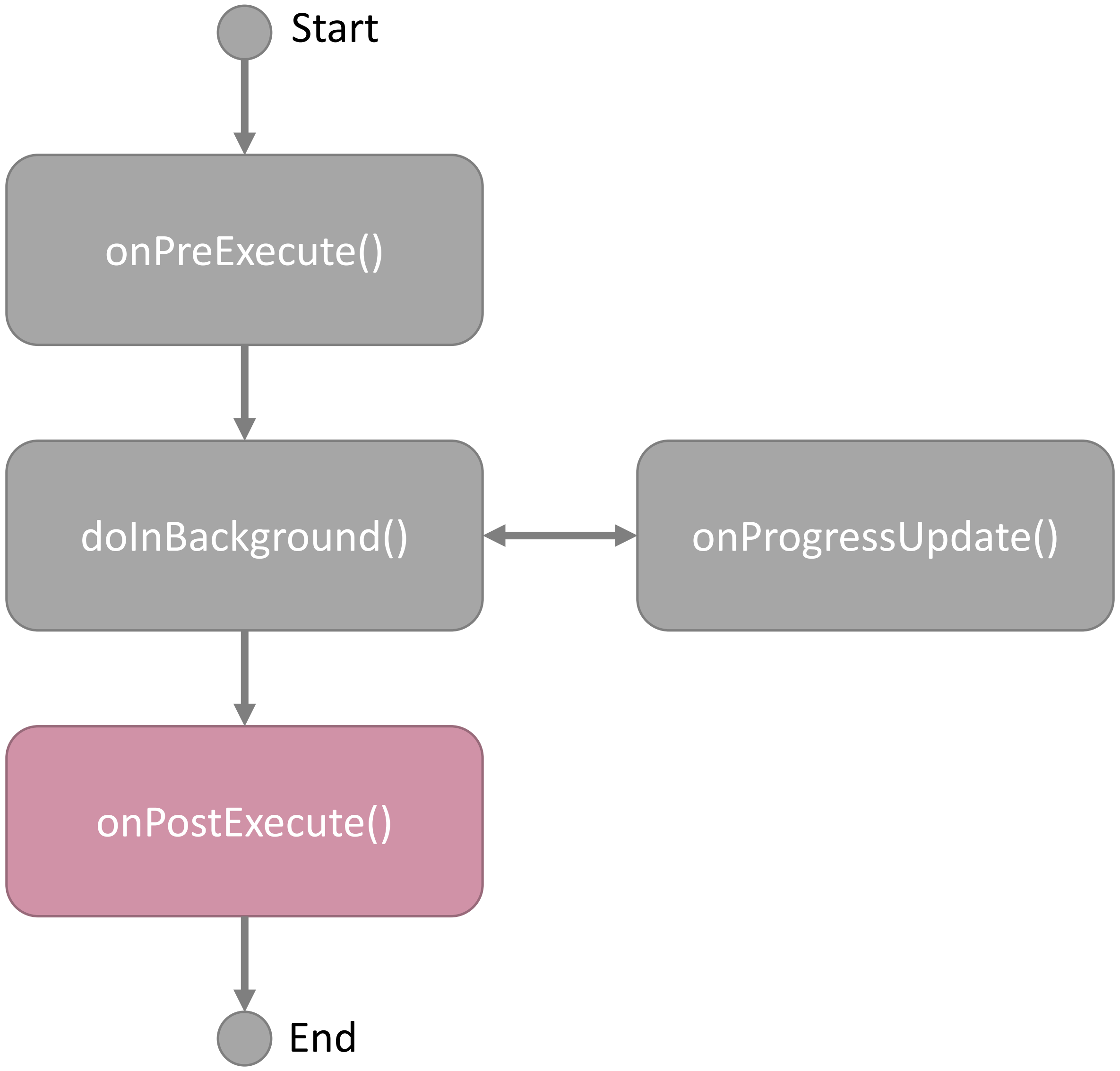
- **NOTE**:
  This function runs on the **UI thread**



AsyncTask

- onPostExecute(): It will be called after doInBackground() finishes its job, here we can:
  - Update UI components using the data received
  - Notify the user that data has been updated (if necessary)

- **NOTE**:
  This function runs on the **UI thread**



AsyncTask

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            if (isCancelled())
                break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

AsyncTask

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            if (isCancelled())
                break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

url list

Volley

- An HTTP library that provides simpler and more efficient method to make networking easier in Android
  - Available through the Android Open Source Project
  - You need to download the code and include it in your own project
  - Similar to AsyncTask, suitable for short operations
  - Not suitable for large downloads, as all responses are kept in memory by Volley

Reference: https://developer.android.com/training/volley/index.html

Using Third-Party Library

- Given that HTTP communication is very common in mobile apps, there are a few widely used third-party libraries for Android that will make your life much easier
- OkHttp: https://square.github.io/okhttp/
- Android Asynchronous HTTP Client: https://loopj.com/android-async-http/

Checking Availability of the Network

Network Availability

- Before using the network to carry out data communication, it is a good practice to first check for the availability of the network
    - On a mobile phone, the network can be unstable
    - The user may have switched off data transmission, or switched to airplane mode
    - By checking the availability of the network, you can prompt the user first before you carry out any network operation

Network Availability

- To be able to check the state of the network, you need to ask for the following permission in the AndroidManifest.xml file

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- To check whether network is available, you can use:

```
ConnectivityManager cm =
    (ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo info = cm.getActiveNetworkInfo();

boolean isConnected = (info != null) && (info.isConnectedOrConnecting());
```

Network Availability

- In addition, you may want to know if the user is connected using mobile data or Wi-Fi. This is useful when you plan to transmit large amount of data.
- The **isActiveNetworkMetered()** function of the **ConnectivityManager** instance indicates if the active network is metered (e.g., mobile data).

```
ConnectivityManager cm =
    (ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE);

boolean isMetered = cm.isActiveNetworkMetered();
```

Loading Multiple Images Using HTTP

Loading Images

- Consider the case when the data requested from server contains both text and URLs to images

```
{
    "data": [
        {
            "title": "News title 001",
            "content": "...",
            "image": "http://www.example.com/images001.jpg"
        },
        {
            "title": "News title 002",
            "content": "...",
            "image": "http://www.example.com/images002.jpg"
        },
        ...
    ]
}
```

Loading Images

```
{
    "data": [
        {
            "title": "News title 001",
            "content": "...",
            "image": "http://www.example.com/images001.jpg"
        },
        {
            "title": "News title 002",
            "content": "...",
            "image": "http://www.example.com/images002.jpg"
        },
        ...
    ]
}
```

For each of these images, you need to download and show the bitmap in an ImageView

Loading Images

- Two third-party Android libraries are particularly useful:
    1. Picasso
       https://square.github.io/picasso/
    2. Glide:
       https://github.com/bumptech/glide

- Both handle downloading the image in a new thread, and update an ImageView on the UI thread
- Handle caching of images
- Transformation of the image before displaying

Project Arrangements

Project

- By the end of the Assignment 4, you will have developed a **working instant messaging app** allowing users to send text messages to one another.
- In the project, your group will develop on top of the instant messaging app by adding some interesting **networking functions**, some examples are:
  - Sending audio messages, images or videos
  - Video broadcasting or streaming
  - Group chat (allowing adding and removing users to a group)
  - Mini-games inside the app
  - Adding friends using NFC
  - …

Project

- Form a group of **2 or 3 students**
- A forum on **Blackboard (China)** (https://cuhk.blackboard.com.cn/) has been set up for searching partners.
- One of the group members should email me with both student IDs before **Feb 28**.
- Think about what kind of extensions you would like to work on in your project (consider the **workload** based on the number of members).

Next Lecture:
Web and Application Servers

Next Lecture

- We will be talking about Web and Application servers in the next lecture. Please get familiar with the following things beforehand:
  - Creating a **Ubuntu VM** in Amazon AWS
    - Reference: https://aws.amazon.com/getting-started/tutorials/launch-a-virtual-machine/
  - Some basic Linux commands (e.g. installing new software)
  - Python
    - References:
      https://docs.python.org/2/tutorial/
      https://www.youtube.com/playlist?list=PLS1QulWo1RIaJECMeUT4LFwJ-ghgoSH6n

End of Lecture 4