CISC 3120
C23: User Agent & Web Server Communication

Hui Chen

Department of Computer & Information Science

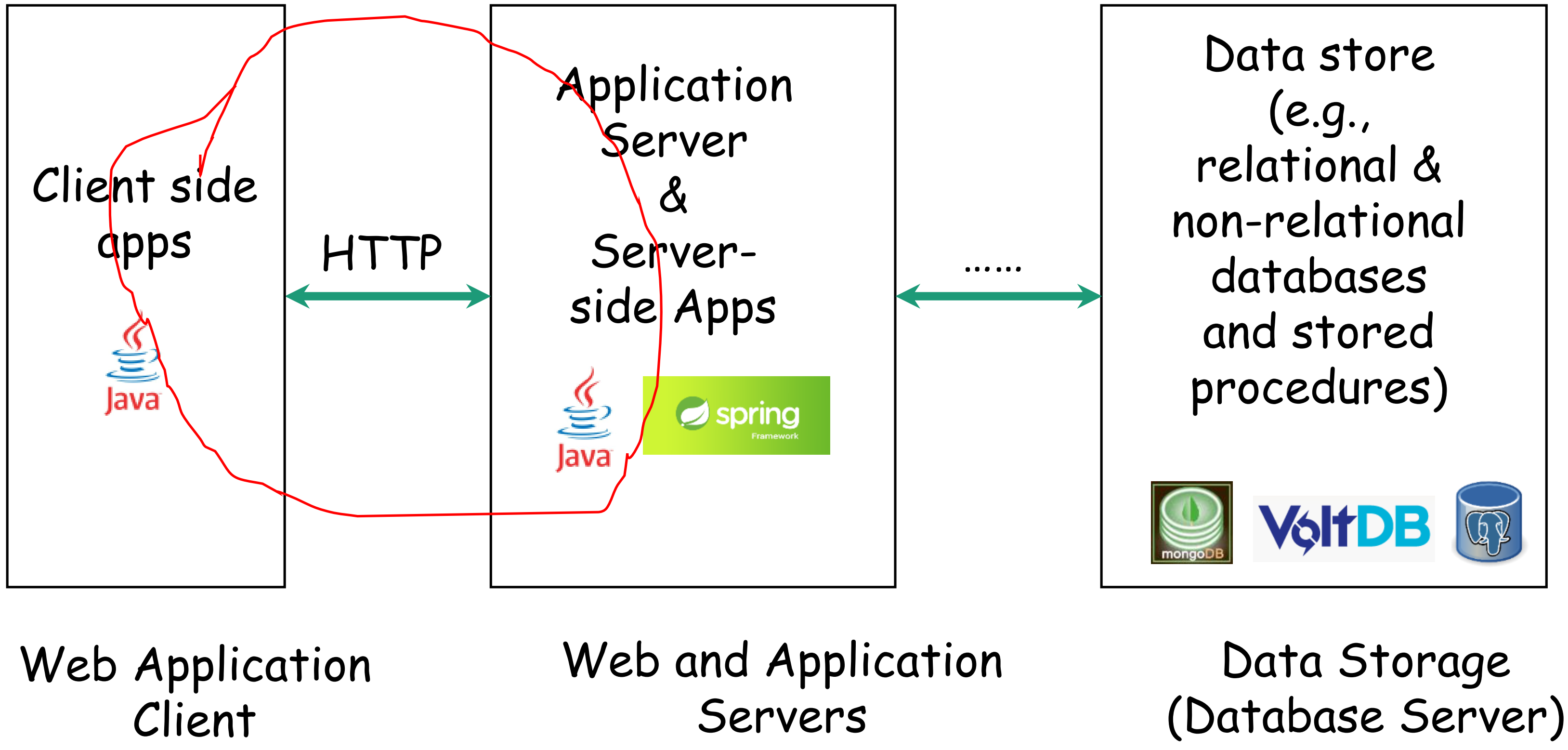
CUNY Brooklyn College

Outline

- Location resources on the Web
    - URI, URL, and URN
- User agent and Web server communications
    - The Hypertext Transfer Protocol (HTTP)
- Programming with simple Web services

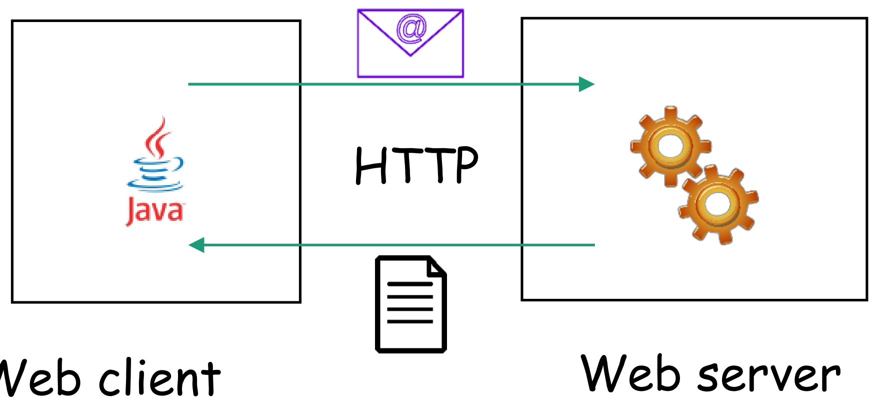
Web Application Architecture

- 3-tier (and n-tier)


Client side apps
Java
HTTP
Application Server & Server-side Apps
Java
spring Framework
……
Data store (e.g., relational & non-relational databases and stored procedures)
mongoDB
VoltDB
Web Application Client
Web and Application Servers
Data Storage (Database Server)

Interaction between Client and Web Server

- Client requests resources from the server
- Server returns resources from the server
- They follow a communication protocol
  - Hypertext transfer protocol (HTTP)



Locating Resources

- In the request, an important field is to identify resource requested
- Uniform Resource Identifier (URI)
- Uniform Resource Locator (URL)
- Uniform Resource Name (URN)

URI, URL, and URN

- Defined in
    - RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax (It obsoletes RFCs 2396, 2732)
- Updated by
    - RFC 6874 and RFC 7320.

URI

- Uniform Resource Identifier
    - A means for identify a resource
    - A sequence of characters from a very limited set
        - The basic Latin alphabet, digits, and a few special characters

URI: Examples

ftp://ftp.is.co.za/rfc/rfc1808.txt

http://www.ietf.org/rfc/rfc2396.txt

ldap://[2001:db8::7]/c=GB?objectClass?one

mailto:John.Doe@example.com

news:comp.infosystems.www.servers.unix

tel:+1-816-555-1212

telnet://192.0.2.16:80/

urn:oasis:names:specification:docbook:dtd:xml:4.1.2

urn:isbn:096139210x

URI: General Syntax

- Syntax: [] indicating optional
  - URI = [scheme:] scheme-specific-part[#fragment]
- Absolute URI
  - when a scheme is specified
- Relative URI
  - when a scheme is not specified

Opaque URI

- An absolute URI whose scheme-specific part does not begin with a slash character ('/')
  - Examples

    mailto:John.Doe@example.com

    news:comp.infosystems.www.servers.unix

    tel:+1-816-555-1212

    urn:oasis:names:specification:docbook:dtd:xml:4.1.2

Hierarchical URI

- Two types
  - An absolute URI whose scheme-specific part begins with a "/" character
  - A relative URI, i.e., a URI that does not specify a scheme
  - Examples

```
ftp://ftp.is.co.za/rfc/rfc1808.txt
docs/guide/collections/designfaq.html#28
../../../demo/jfc/SwingSet2/src/SwingSet2.java
```

Hierarchical URI: General Syntax

- Syntax
  - [scheme:][//authority][path][?query][#fragment]
  - Example

```
foo://example.com:8042/over/there?name=ferret#nose
\_/   \______________/\_________/ \_____ __/ \__/
 |           |             |           |        |
scheme   authority        path       query  fragment
```

  - Sometimes a hierarchical path or a part of it resembles a file system path, but NOT imply that the resource is a file or that the URI maps to an actual filesystem pathname.

Net Authority

- When authority is started with "//"
- Server-based authority
    - [user-info@]host[:port]
- Example: an URI with server-based net authority
    - foo://johnsmith@example.com:8042/over/there?name=ferret#nose

Query in Hierarchical URI

- Recall the syntax
  - [scheme:][//authority][path][?query][#fragment]
  - Example
    - foo://example.com:8042/over/there?name=ferret&class=senior&major=cisc#gpa
  - The query in the example is
    - name=ferret&class=senior&major=cisc
  - Query components are in the form of "key=value" pairs and often separated by “&”.

Questions?

- Syntax of URI
- Opaque URI
- Hierarchical URI
    - Scheme
    - Authority
    - Path
    - Query
    - Part

Characters in URI

- Only these characters are permitted in various parts of URI
  - alpha: the US-ASCII alphabetic characters, 'A' -'Z' and 'a' - 'z'
  - digit: the US-ASCII decimal digit characters, '0' - '9'
  - alphanum: all alpha and digit characters
  - unreserved: all alphanum characters and _ - ! . ~ ' ( ) *
  - punct: , ; : $ & + =
  - reserved: all punct characters and ? / [ ] @
  - escaped: escaped octets, i.e., triplets consisting of the percent character ('%') followed by two hexadecimal digits ('0'-'9', 'A'-'F', and 'a'-'f')
  - other: Unicode characters that are not US-ASCII, not control, and not space (e.g., ¡hola!, 你好, مرحبا, שלום)

Encoding and Decoding

- Escaped octets can appear in the user-info, path, query, and fragment components
  - Replace character by an escape sequence
    - To encode non-US-ASCII characters
    - To quote characters that are otherwise illegal in a component.
  - Example
    - € ('\u20AC'), encoded as %E2%82%AC
    - Space character, encoded as %20

Questions?

- Permitted characters in URI
- Encoding and decoding

Dealing with URI in Java

- The java.net.URI class
  - Construct instances and retrieves various parts
    - scheme, authority, port, user info, path, query, fragment
  - In hierarchy URI, "." and ".." represent the current and the parent in the hierarchy
    - Normalization: removing unnecessary "." and ".."
    - Resolution: resolve one URI against another
    - Relativization: inverse of resolution
  - Programming examples

Questions

- Dealing with URI in Java
  - Construct instances (how to?)
  - Normalization, resolution, and relativization (how to, what do they mean?)
  - Identities (what have we experimented?)
- How about encoding and decoding?

URL and URN

- Uniform Resource Locator
  - A subset of URIs, specify the primary access mechanism in the URI (e.g., its network "location").
  - Example
    - http://www.ietf.org/rfc/rfc2396.txt
- Uniform resource name
  - A URI that does not specify how to retrieve them
  - Example

    urn:oasis:names:specification:docbook:dtd:xml:4.1.2

    urn:isbn:096139210x

URI and URL in Java

- java.net.URI
    - An instance of the class is a URI instance is little more than a structured string
        - To support comparison, normalization, resolution, and relativization in syntactic, scheme-independent fashion
- java.net.URL
    - represents two pieces of information
        - (1) the syntactic components of a URL, and (2) the information required to access the resource that it describes
    - An URL must be an absolute URI (otherwise?)
    - A stream handler is always established for a URL
        - URL encompasses network I/O operations of looking up the host and opening a connection to the specified resource

URL Decode and Encode

- URI's toURL method does encoding
  - http://www.example.com/student/¡hola! → http://www.example.com/student/%C2%A1hola!
- Two classes
  - java.net.URLDecoder and java.net.URLEncoder

Experimenting with URL in Java

- Create an URL from an URI instance
  - If an URL instance represents a URL
  - java.net.URI's toURL() method
- URL operations
  - Besides obtain syntactic components
    - [scheme:][//authority][path][?query][#fragment]
  - Get the file name
  - Open a connection
  - Open a stream (any resource)
- Programming examples

Questions?

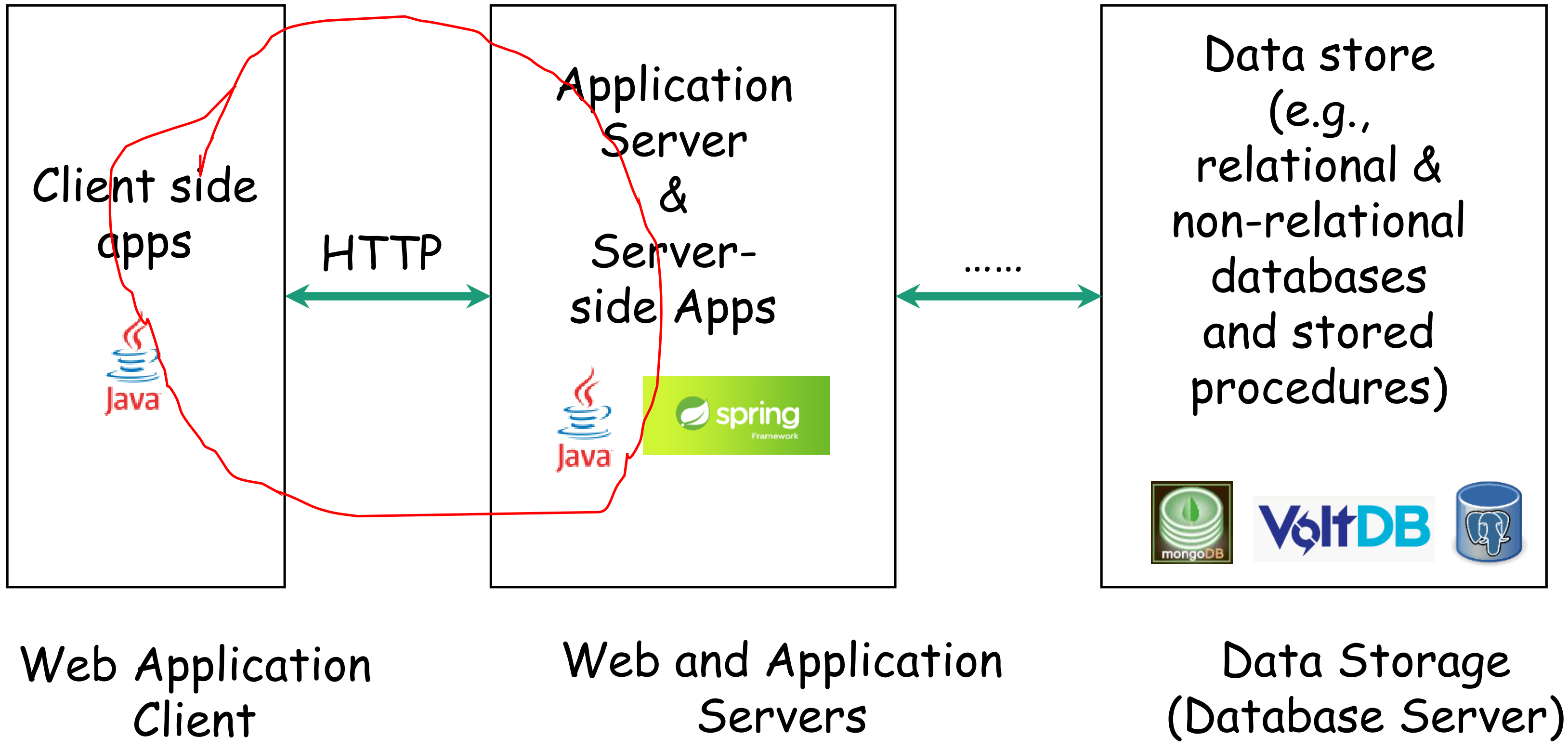
- URI, URL, and URN?
- What to do with URL?
  - Creating an URL instance
  - Get file info
  - Open connection
  - Stream I/O

Discussion about URL

- How does the URL instance open a stream for the resource in the example program?
- The URL is
  - http://www.brooklyn.cuny.edu/web/home.php

Web Application Architecture

- 3-tier (and n-tier)

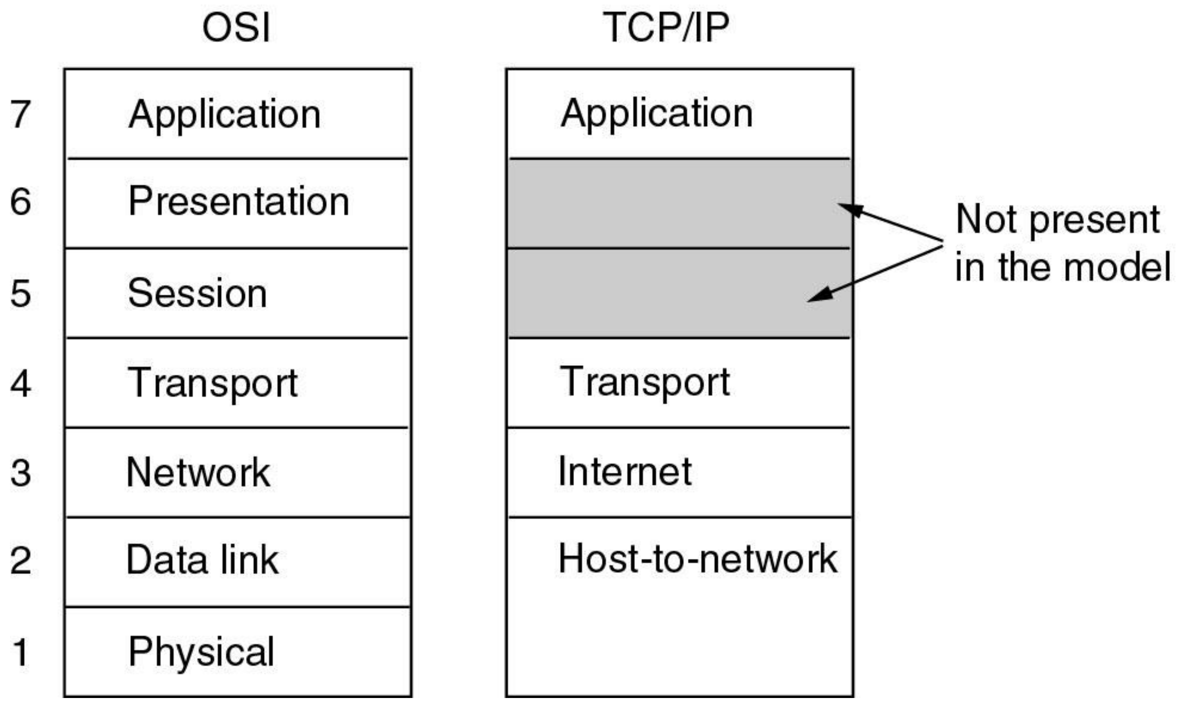


Client side apps
Java
HTTP
Application Server & Server-side Apps
Java
spring Framework
……
Data store (e.g., relational & non-relational databases and stored procedures)
mongoDB
VoltDB
Web Application Client
Web and Application Servers
Data Storage (Database Server)

HTTP

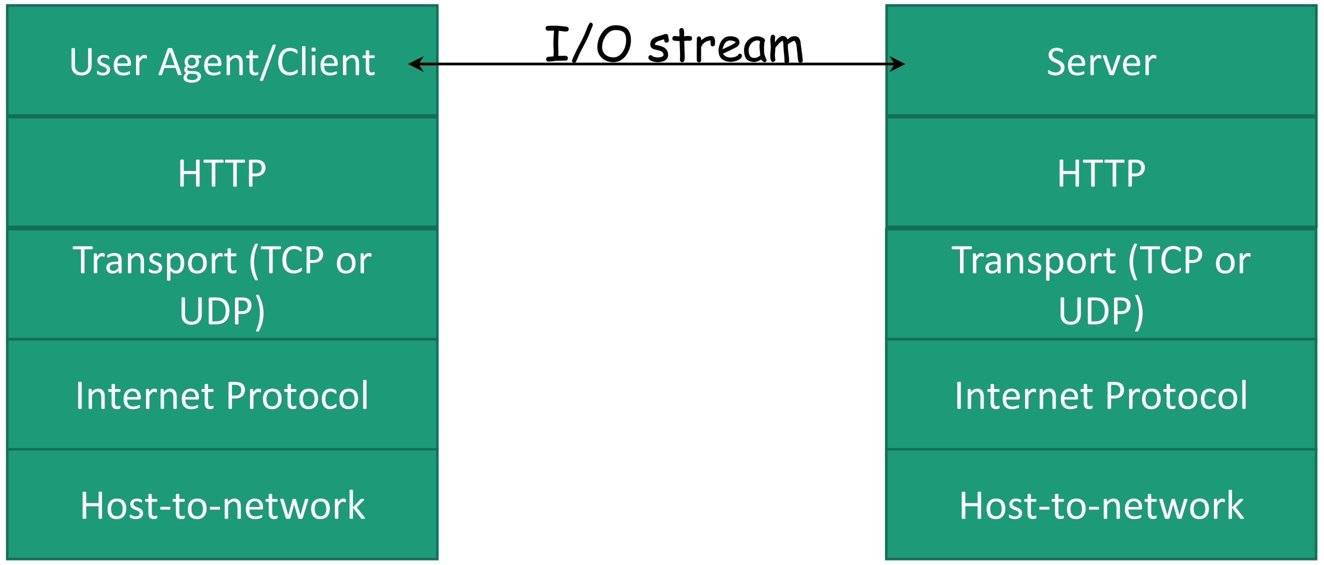
- Hypertext Transfer Protocol
  - Simple request-response protocol layered on TCP/IP
  - Where does it belong in the OSI 7-layer model and the TCP/IP model?

HTTP: An Application Layer Protocol



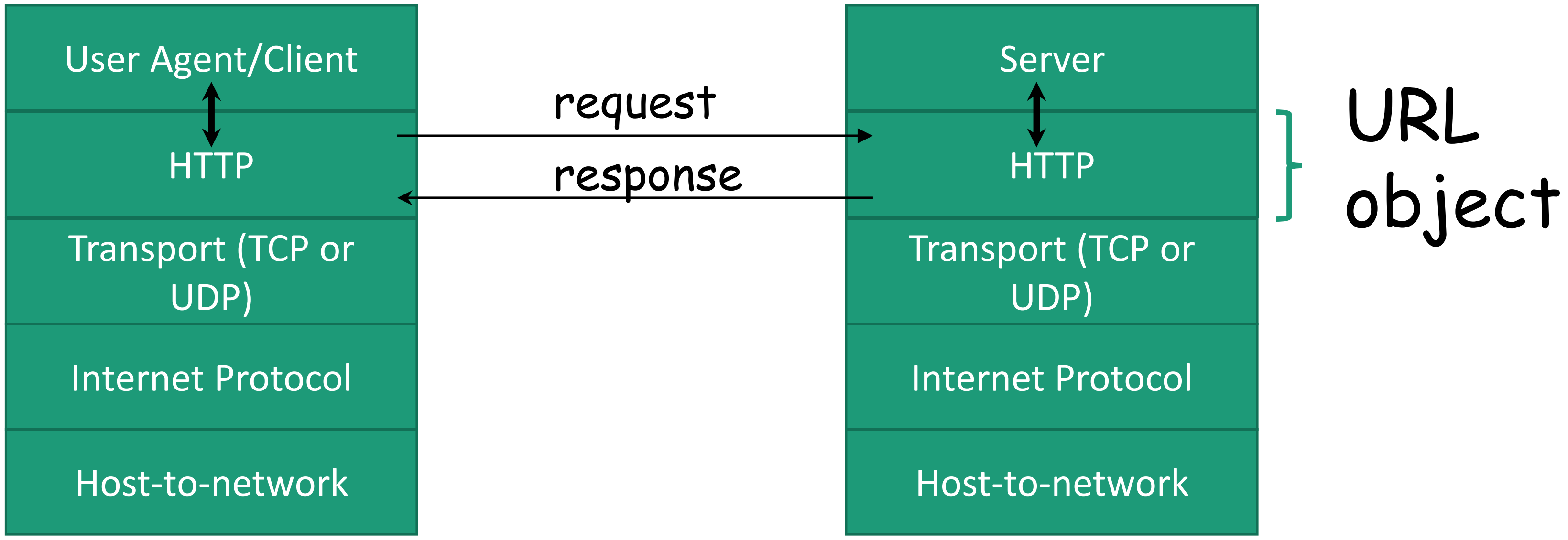
Two Communicating Apps

- Two applications communicates with each other



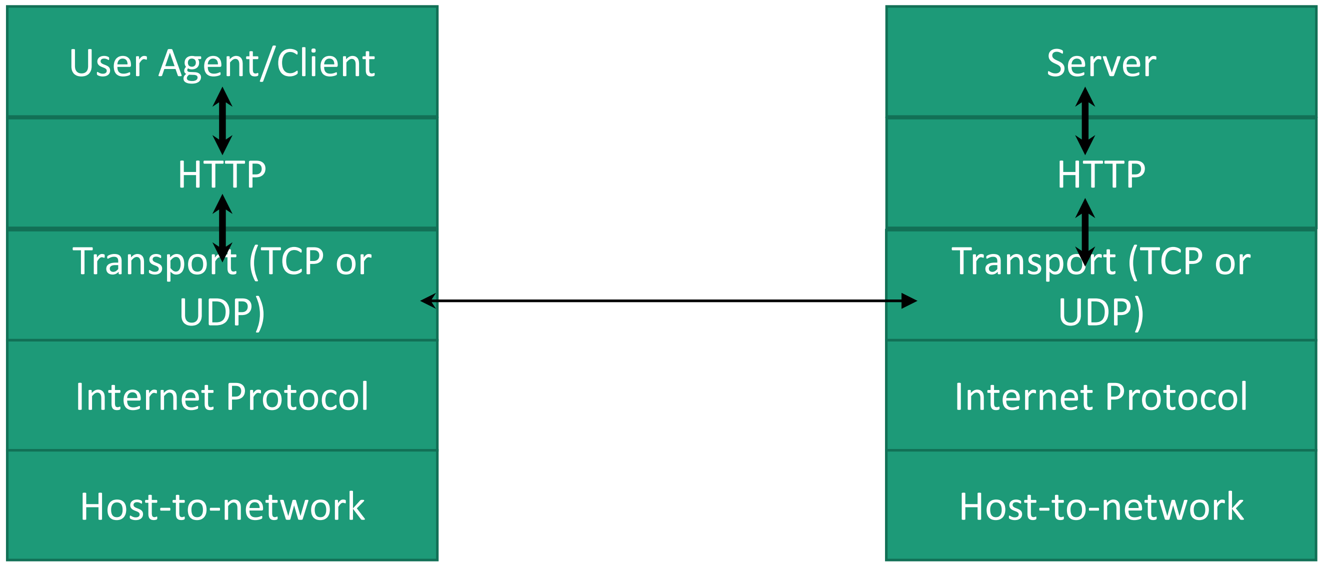
Communication with HTTP

- Two applications communicates with each other via HTTP



HTTP over Transport Layer

- Two applications communicates with each other via HTTP



Testing the Understanding

- Implement an application
  - Construct a request
    - How does it look like?
  - Open a TCP connection to the server
  - Send the request
  - Receive a response
    - How do we make sense of the response?
  - Close the TCP connection

Questions?

- The big picture
  - Communication applications over HTTP
  - HTTP over Transport layer

HTTP Message Exchange

- A typical scene involves a request and response cycle
  - A client establishes a “connection” to the sever
  - The client sends a HTTP request along the connection to the server
  - The server replies the client with a response
  - The client reads from the “connection” the response from the web serve

Example: HTTP/1.1 Request

- Header and Body

```
GET /index.html HTTP/1.1
User-Agent: Java Web Client
Host: localhost:61235
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive


```

Request method, URI, protocol version

Two blank lines

header

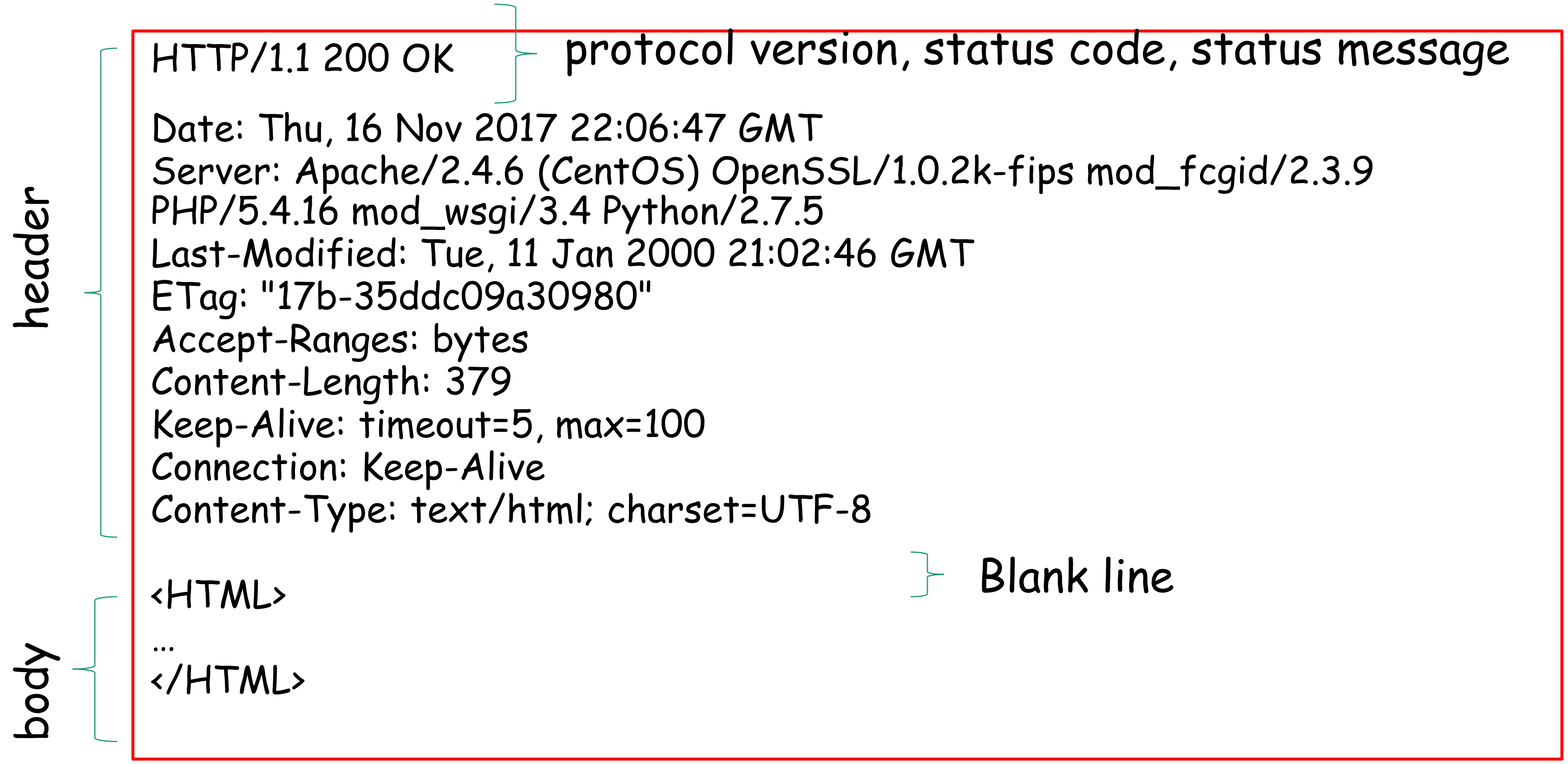
body

HTTP Request Methods

- GET
  - fetch a URL
- HEAD
  - fetch information about a URL
- PUT
  - store to an URL
- POST
  - send form data to a URL and get a response back
- DELETE
  - delete a URL
- Most frequently used methods are GET and POST

Example: HTTP/1.1 Response

- Header and body



```
HTTP/1.1 200 OK

Date: Thu, 16 Nov 2017 22:06:47 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips mod_fcgid/2.3.9
PHP/5.4.16 mod_wsgi/3.4 Python/2.7.5
Last-Modified: Tue, 11 Jan 2000 21:02:46 GMT
ETag: "17b-35ddc09a30980"
Accept-Ranges: bytes
Content-Length: 379
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<HTML>
…
</HTML>
```

HTTP Response Status Code

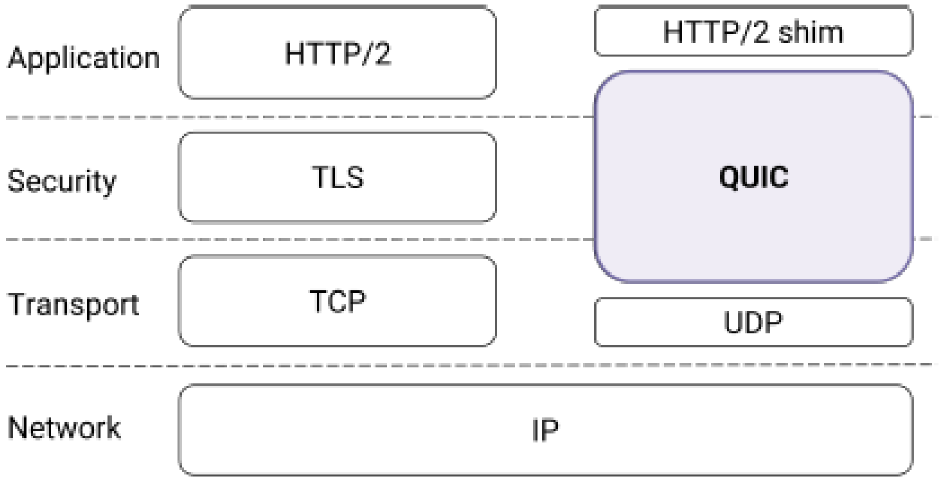
- Constants in java.net.HttpURLConnection class
- Also: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
- 100 ~ 199: informational
- 200 ~ 299: successful
  - 200: OK
- 300 ~ 399: redirection
  - 301: Moved Permanently
- 400 ~ 499: client error
  - 404: Not Found
- 500 ~ 599: server error
  - 500: Internal Server Error
  - 503: Service Unavailable

HTTP Evolution

- HTTP 0.9 – 1.0: initial development; 1991 – 1996
  - Allows only one outstanding request at a time on a given TCP connection
- HTTP/1.1: standardized in 1997
- HTTP/2: standardized in 2015
  - Aimed reduce latency
  - Allows interleaving requests & responses on the same connection, reduces header size, supports prioritization of requests

Some Recent Development

- Web becomes an application platform
- Secure Web traffic becomes dominant
- Handshake latencies
    - TCP: 1 round-trip delay; TLS: 2 round-trip delay



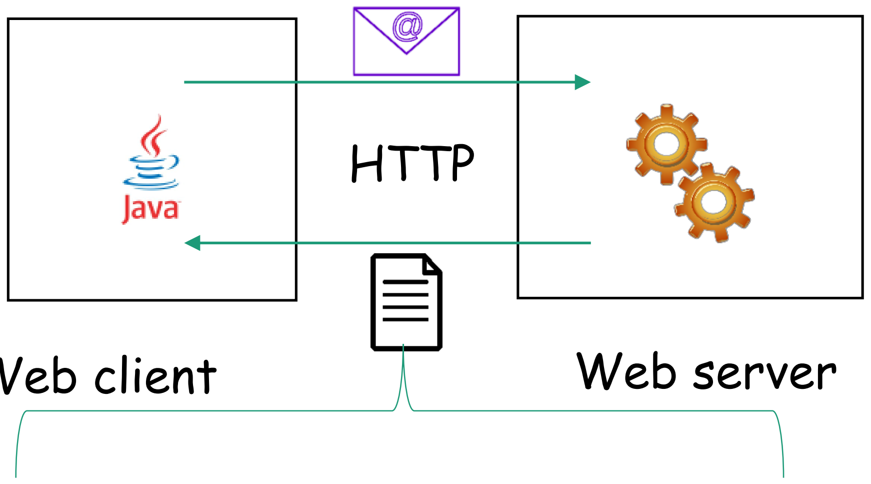
[Langley et al., 2017]

Question?

- Understand HTTP request and response
  - HTTP/1.1 request and response
- Evolving to HTTP/2
- Recent development to reduce latency for Web applications

Web Applications?

- What's in the response?



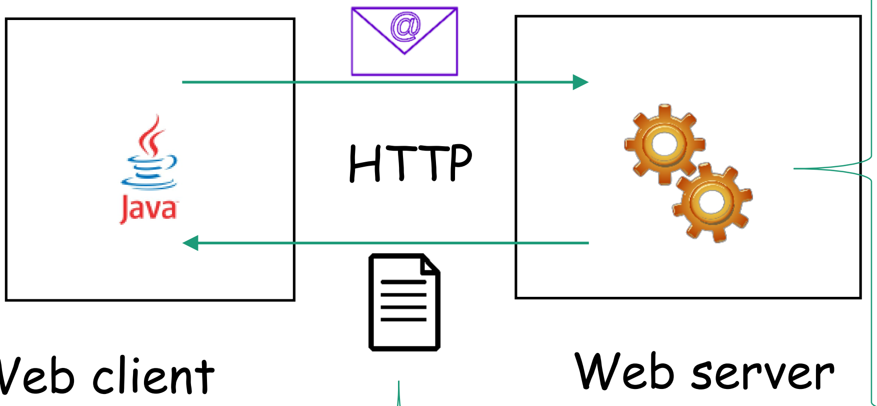
HTML document, consumed by
- Humans (primary)
- Programs (secondary)

JSON/XML, consumed by
- Programs (primary)
- Humans (secondary)

Static/dynamic websites ⟷ Web applications

Access Service on the Web

- What's in the response?



Web Service: Programs written for building Web applications, residing on the Server

Output

HTML document, consumed by
- Humans (primary)
- Programs (secondary)

JSON/XML, consumed by
- Programs (primary)
- Humans (secondary)

Static/dynamic websites ⟷ Web applications

Web Services: Examples

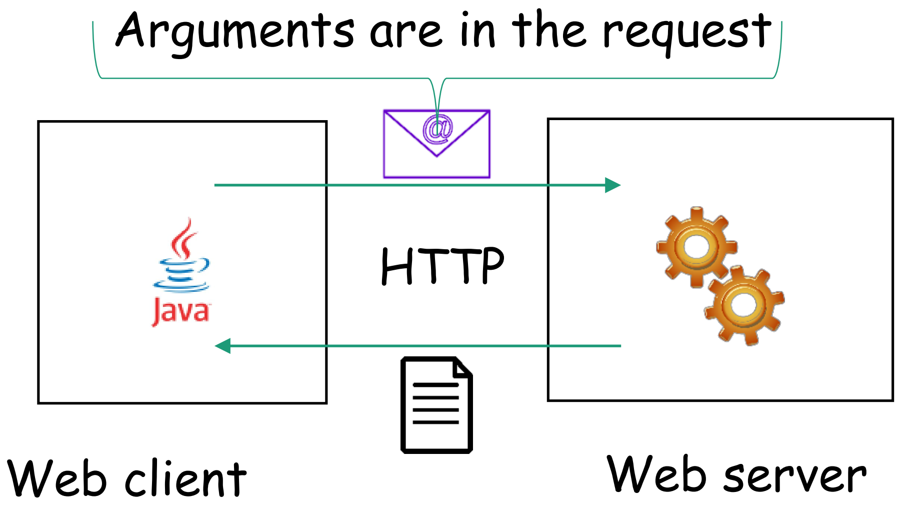
- Web API: interfaces of Web service programs, expressed as URLs
- Stack Exchange
  - https://api.stackexchange.com/docs
- Google API
  - Maps API: https://developers.google.com/maps/documentation/geocoding/start
- Facebook API
  - wit.ai: https://wit.ai/docs/http/20170307
- The NYC Open Data API
  - Open311: http://wiki.open311.org/GeoReport_v2/
- Stock Price
  - Alpha Vantage: https://www.alphavantage.co/documentation/

Let's Build a Small Application

- Auto-fill address by zip code?
  - Official from USPS: https://www.usps.com/business/web-tools-apis/documentation-updates.htm
  - A very simple Web API by Thomas Schultz and Josh Strange
    - http://ZiptasticAPI.com/

Passing Arguments

- Calling a method, passing arguments
  - How do we realize it here?



Passing Arguments via URL

- URL syntax
  - $\text{scheme://authority}\overbrace{\underset{\substack{\downarrow\\ \text{Path variables}}}{\text{[path]}}\,\underset{\substack{\downarrow\\ \text{Query parameters}\\ \text{(request parameters)}}}{\text{[?query]}}\,\text{[\#fragment]}}^{\text{Contains arguments}}$

Passing Argument via Request Body

- Request

```
POST /index.html HTTP/1.1
User-Agent: Java Web Client
Host: localhost:61235
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive


......
```

header

body

Request method, URN, protocol version

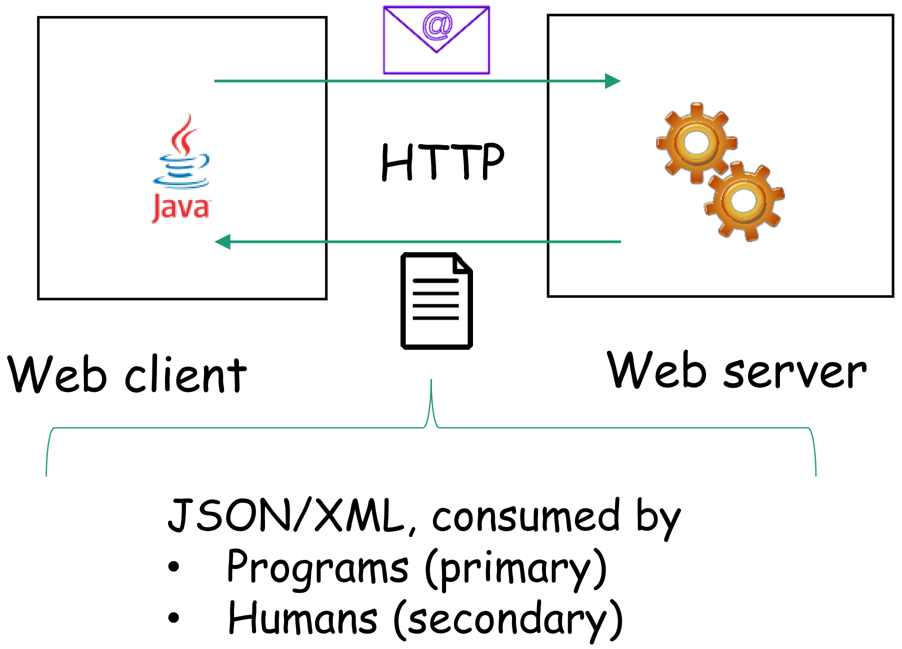
Two blank lines

Contains arguments (request parameters)

Return Value in the Response

- Web services typically returns values in either JSON or XML format in the response body
- JSON: an example
  - A key-value list

```
{
    "country":"US",
    "state":"NY",
    "city":"BROOKLYN"
}
```



Let's Program!

- JavaFX user agent
- Web service by ZiptasticAPI.com

Questions?

- Concept of Web applications
- Locating resources on the Web
    - URL and URI
- User agent and Web server communications
    - HTTP, request, response, status
- Concept of Web APIs and URL
- A simple Web application
    - How do we "invoke" a method on the Web?
    - How do we "pass the arguments"?
    - How do we receive "the return value"?