CSE 1710

Lecture 20

Net-Centric Programming, Part I

Learning Outcomes

- understand net-centric functionality in terms of client and server roles
- see how the Internet Protocol Suite is an example of layered abstraction
- distinguish between the WWW and the Internet
- Understand what the `URL` class encapsulates
- Understand what the `URLConnection` encapsulates
- programmatically get static content from a URL

Review: what does it mean for a method *to block*?

- when a method is invoked, each statement in the body of the method is invoked in sequence
  - somewhere in the body of the method, a statement is waiting for "something" to happen
  - until this "something" happens, the method **blocks**
- the complete invocation of the method depends on some outside event, which may or may not happen in a timely fashion

Examples L20App1

`readLine()` from `Scanner`

`getResponseCode()` in `HttpURLConnection`

Review: reading from a `File` object

Recall sec 5.3.2 "File I/O" (p.199)

L20App2

Review: the `instanceof` operator

Recall sec 3.2.4 “Relational Operators” (p.110)

```
<       <=      >       >=      ==      !=
```

There was also the following operator:

```
instanceof
```

```
boolean test = x instanceof C;
```

the expression evaluates to true iff either:

- the object reference `x` references an instance of class `C` or
- the object reference `x` references an instance of a subclass of `C`

L20App3

The Big(ger) Picture

- HTTP is the protocol used to access services concerning remote html files
- It is an *application layer* protocol
- HTTP is but one of several possible application layer protocols
- The application layer protocol is part of the Internet protocol suite (aka TCP/IP)

The Internet Protocol Suite

- Physical layer
- Data link layer
- Network layer
- Transport Layer
- Application Layer

- each layer has its own specific task to perform
- each task has its own set of issues, its own specific data unit
- the suite is an example of layered abstraction as a strategy to confront complexity
  - Each layers hides the details under it. It appears as a service to the layer above it

The Internet Protocol Suite

- Physical layer
  - layer deals with: data bits
  - task: how to encode 0 and 1 as an analog signal, how to transmit that one bit of data from a computer's Network Interface Card (NIC) to the transmission medium (e.g., copper wire, the air, etc)
  - protocols include: Ethernet, WiFi (aka IEEE 802.11), FireWire
- Data link layer
  - layer deals with: frames
  - task: transmit one frame from one node to another on a LAN
  - protocols include: Ethernet, WiFi (aka IEEE 802.11), others (not FireWire)
- Network layer
  - layer deals with: datagram (aka packet)
  - task: transmit packets from one node to another on a LAN
  - protocols include: IP, others

The Internet Protocol Suite

- Transport Layer
  - layer deals with: segments (to/from PORT numbers)
  - task: transmit messages (segment) from a process running on a node in one LAN to one running on a node in another LAN
  - protocols include: TCP, others
- Application Layer
  - layer deals with: messages
  - task: provide services to user (in the form of message sending/ receiving)
  - protocols include: HTTP, DNS, FTP, SSH, TELNET, SMTP, SIP, many others

Why is the WWW ≠ The Internet?

- The Web is just a portion of the Internet
  - It is the subset of the Internet Protocol Suite that is concerned with HTML pages
- The Web went “live” on Aug 6, 1991
  - proposed two years earlier (by Tim Berners-Lee, who was working at CERN at the time)
- The Internet existed way before the Web
  - prior to TCP/IP (1970’s), ARPANET was created (1960’s)
- There is more to the Internet than web pages
  - bulk of Internet is used for peer-to-peer file sharing, not for client-server html sending/receiving

The DNS Application Layer Service

a naming system that maps names to IP addresses

e.g., maps `130.63.92.30` to `cse.yorku.ca`

- why do we need this?
  - IP addresses are numeric, not easy for people and applications to use
    - difficult to remember, difficult to associate with meaning
  - IP addresses can be reassigned or otherwise change

Components of a HTTP-specific URL

- e.g., `http://cse.yorku.ca:80/course/1710/index.html`

- the URL represents a reference to an object (html file) that is remote (lives on the web-server "cse.yorku.ca")
- The reference has the following components:
  - the protocol (http)
  - a path: the location of the information
    - the information needs to "live" somewhere on the host machine
    - static information is already composed, formatted as html, and stored in a file
    - dynamic information gets composed on-the-fly and then gets formatted as html; it exists only as a run-time entity
  - (optionally) a port, if something other than the default of port 80 is needed

Components of a URL (in general)

- e.g., `ftp://ctan.org`

- still represents a reference to a remote object, but the object will be something other than html-formatted text
  - It will consist of a protocol
    - ftp, smtp, etc
  - a path: the location of the information
    - the information needs to "live" somewhere on the host machine
    - static vs dynamic:
      - the information is already composed, formatted according to protocol, and stored in a file
      - the information gets composed and then gets formatted in a protocol-specific way; it exists only as a run-time entity
  - (optionally) a port: the port number to which the TCP connection is made on the remote host machine
    - recall the Transport Layer

What services does the `URL` class provide?
What does an `URL` object encapsulate?

- The class `URL` is found in the `java.net` package
- The class provides a constructor that accepts a string
- encapsulates:
  - the protocol that is being used
  - all the detail about the Internet Protocol Suite that is needed to actually obtain services according to that particular protocol

What do all protocols have in common?

- All protocols provides a means to establish a connection to the remote object
- The connection to the remote object *is abstracted away* from the URL itself
  - it encapsulated by another service, `URLConnection`
  - in the particular case of the HTTP protocol, the connection is encapsulated by `HttpURLConnection`
  - the particular method is `openConnection()`

L20App4

from this point forward, we will limit our discussion to URLs using the http protocol