

# CSE 1710

## Lecture 21

### *Net-Centric Programming, Part II*

Part2

#### **Learning Outcomes**

- Understand and describe the basics of the HTTP protocol
- Use the `URL` and `URLConnection` classes to:
  - to instantiate useful objects
  - to retrieve content from web servers
    - Use string processing to manipulate query strings
- Understand the concept of a class hierarchy
  - run-time checking using `instanceof`
- Use the `HttpURLConnection` class to examine the request and response messages

## The Basics of HTTP

- HTTP stands for Hypertext Transfer Protocol
- HTTP is used to transmit resources, not just files.
- A resource is some chunk of information
  - Something that can be identified by a URL (it's the R in URL).
  - Examples of resources:
    - files
    - a dynamically-generated query result
    - the output of a CGI script
    - a document that is available in several languages.

3

## The Basics of HTTP

- HTTP uses a model in which there is a client and a server role (the “client-server” model):
  - An HTTP **client** opens a connection and sends a request message to an HTTP **server**
  - A HTTP **server** then returns a response message to the **client**, usually containing the resource that was requested
  - After issuing the response, the **server** closes the connection.

4

## The Basics of HTTP

- an HTTP transaction is defined as:
  - a single request from a client and the corresponding response from the server
- it often happens that a given pair of a client and a server pairing will have several transactions in sequence
  - however, **no connection information** is maintained between transactions
  - this is called “stateless” and is what is meant by “http is a stateless protocol”

5

## The Basics of HTTP

- There are two types of messages:
  - request messages
  - response messages
- Both kinds of messages consist of:
  - an initial line
  - zero or more header lines
    - e.g., the “Date” field represents the date and time at which the message was originated
  - a blank line
  - (optionally, but not necessarily) a message body (aka “payload”)
    - e.g. a file, query data, or query output

6



## Response Codes

100 series

Sessional update from server.

200 series

Success!

300 series

Redirect.

400 series

Client error.

500 series

Server error.

**For full detail, you can look at the full specification at:**

<http://kb.globalscape.com/KnowledgebaseArticle10141.aspx>

9

## Revisit L20App4

Recap each of the statements

10

## Issues with L20App4

We see that L20App4 effectively performed a single transaction

How and Where?

- there was a request message –the `openConnection()` method causes the instantiation of a `URLConnection` object
  - the `URLConnection` object, upon instantiation, attempts to establish contact to the server
  - things could go wrong, for instance an `java.net.UnknownHostException` may be thrown
  - if the connection is established, then the `URLConnection` object will issue the request message
- there was a response message
  - the server will issue a response, which the `URLConnection` object will capture
  - things could go wrong with the connection and an exception will be thrown

How can we examine the specifics of the request and response messages?

We have a `URLConnection` object, L21App1  
but it is a specific version of this: a `HttpURLConnection` 11

## Issues with L20App4

How can we treat the `URLConnection` object as a `HttpURLConnection` object?

Approach #1: L21App1

- cast the object at run-time
- vulnerable if the connection object is not actually an http connection
- this example points out the difference between early binding (p.103) and late binding (at run-time)

Approach #2:

- cast the object at run-time, but do so only conditionally L21App2

## How to check the return code

The HttpURLConnection class offers the following services:

```
String : getRequestMethod()           L21App3  
int    : getResponseCode()  
String : getResponseMessage()
```

13

## Query Strings

Let's have a look at <http://www.cse.yorku.ca/~roumani/jba/ase/>

14