Distributed Systems
Lecture 6

URLConnections and the http headers

Joseph Phillips

Last modified 2019 May 13

Reading

- Chapter 7 “*URLConnections*” of Elliotte Rusty Harold “*Java Network Programming: 4th Ed.*”

Topics

- Java URLConnection class
- Client to Server http header lines
- Server to Client https header lines

Motivation

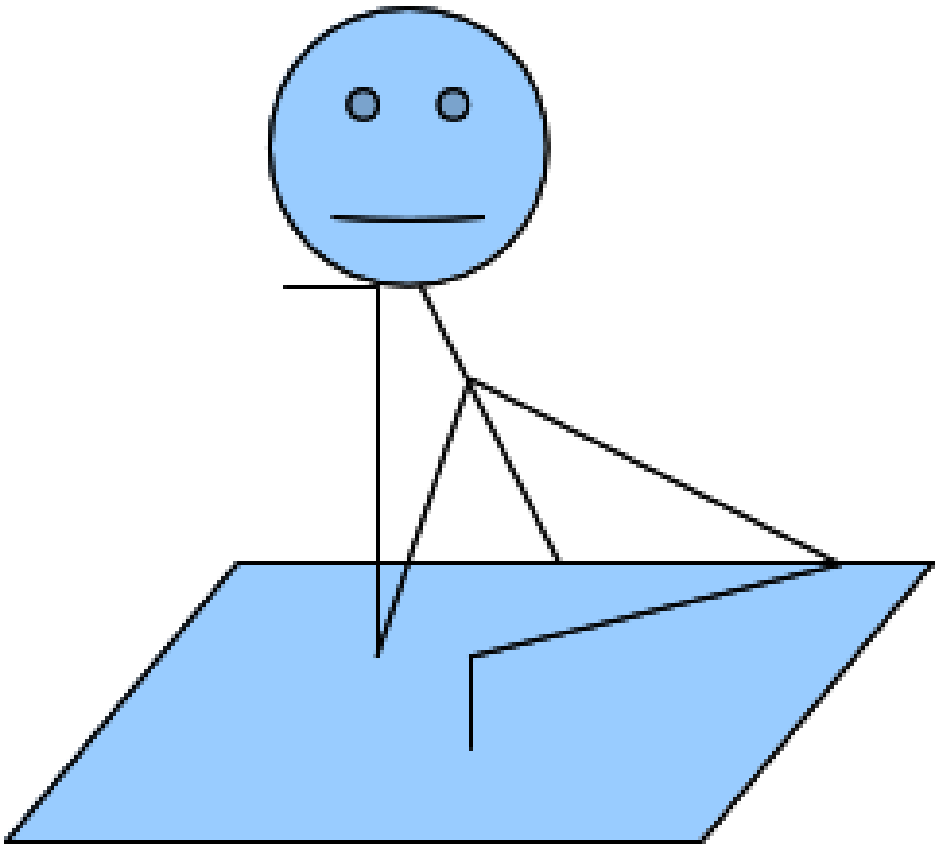
Question: "***So, what does the client get from the server?***"

Answer: "***The resource at the URL, and the header!***"

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=9186D7D17844A85C7B2AE488CE6EB489; Path=/jriely/; HttpOnly
Accept-Ranges: bytes
ETag: W/"15711-1552589539000"
Last-Modified: Thu, 14 Mar 2019 18:52:19 GMT
Content-Type: text/html
Content-Length: 15711
Date: Mon, 06 May 2019 07:55:33 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
  . . .
```

The header?!?

- Isn’t that boring?!
- Shouldn’t we go right to the resource if it’s available?

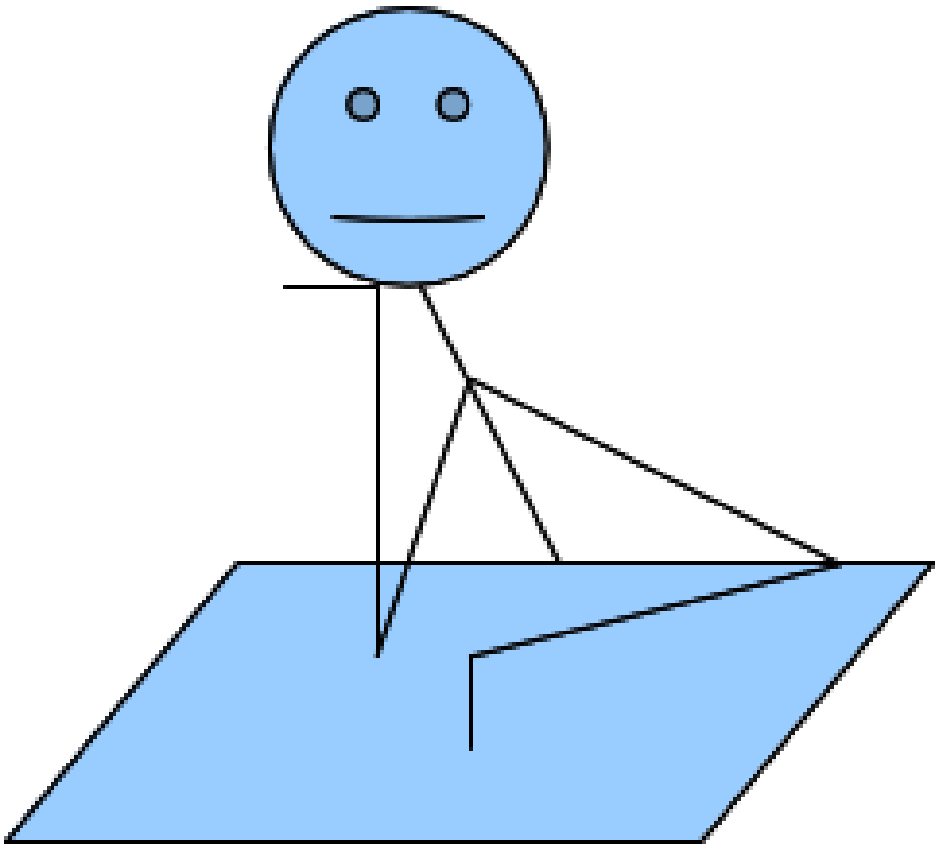


A second look at the header

- True, we want the resource, but the header tells us how to interpret the resource.
  - Content-type
    - e.g. text/html, image/png
  - charset
    - e.g. UTF-8

Getting the header

- Okay, so how do we get the header?



Getting the data: Java

```
public string
URLConnection.getHeaderFieldKey(int index)
```

- Gets header field name
- Returns `null` for index == 0

```
public string
URLConnection.getHeaderField(int index)
```

- Gets header field value
- When index == 0 tells the ***request method***, *e.g*. GET, path, and http version number)
- Returns `null` when there are no more header fields

Getting the data: Java

```
import java.io.*;
import java.net.*;

public class AllHeaders
{
  public static void main      (String args[])
  {
    for  (int i = 0;  i < args.length;  i++)
    {
      try
      {
        URL     u       = new URL(args[i]);
        URLConnection
              uc      = u.openConnection();

        for  (int j = 0;  ;  j++)
        {
          String header= uc.getHeaderField(j);
          if  (header == null)
            break;

          System.out.println
          (uc.getHeaderFieldKey(j) + ": " + header);
        }
      }
      catch  (MalformedURLException ex)
      {
        System.err.println
        (args[0] + " is not a parseable URL");
      }
      catch  (IOException ex)
      {
        System.err.println(ex);
      }
      System.out.println();
    }
  }
}
```

Your Turn!

Write a program to get the content-type!

Getting the data: C (curl)

Yes, nastier than Java

Same approach as getting resource with `CURLOPT_WRITEFUNCTION` and `CURLOPT_WRITEDATA`

- `curl_easy_setopt(CURL* curl, CURLOPT_HEADERFUNCTION, WriteMemoryCallback);`
  - Same type of write-back function as for resource
- `curl_easy_setopt(CURL* curl, CURLOPT_HEADERDATA, void* userPtr);`
  - Same user data-structure as for resource

Getting the data: C (curl)

```
//  fromServerHeaderExaminer.c
//  Modified from cookie_interface.c
/
************************************************************
*************
 *
 *  Project                     ___| | | |  _ \| |
 *                             / __| | | | |_) | |
 *                            | (__| |_| |  _ <| |___
 *                             \___|\___/|_| \_\_____|
 *
 * Copyright (C) 1998 - 2018, Daniel Stenberg,
<daniel@haxx.se>, et al.
 *
 * This software is licensed as described in the file COPYING,
which
 * you should have received as part of this distribution. The
terms
 * are also available at
https://curl.haxx.se/docs/copyright.html.
 *
 * You may opt to use, copy, modify, merge, publish,
distribute and/or sell
 * copies of the Software, and permit persons to whom the
Software is
 * furnished to do so, under the terms of the COPYING file.
 *
 * This software is distributed on an "AS IS" basis, WITHOUT
WARRANTY OF ANY
 * KIND, either express or implied.
 *

************************************************************
*************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <curl/curl.h>
```

```
struct MemoryStruct {
  char *memory;
  size_t size;
};

static size_t
WriteMemoryCallback(void *contents, size_t size,
size_t nmemb, void *userp)
{
  size_t realsize = size * nmemb;
  struct MemoryStruct *mem = (struct
MemoryStruct *)userp;

  char *ptr = realloc(mem->memory, mem->size +
realsize + 1);
  if(ptr == NULL) {
    /* out of memory! */
    printf("not enough memory (realloc returned
NULL)\n");
    return 0;
  }

  mem->memory = ptr;
  memcpy(&(mem->memory[ mem->size]), contents,
realsize);
  mem->size += realsize;
  mem->memory[ mem->size] = 0;

  return realsize;
}
```

Getting the data: C (curl), cont'd

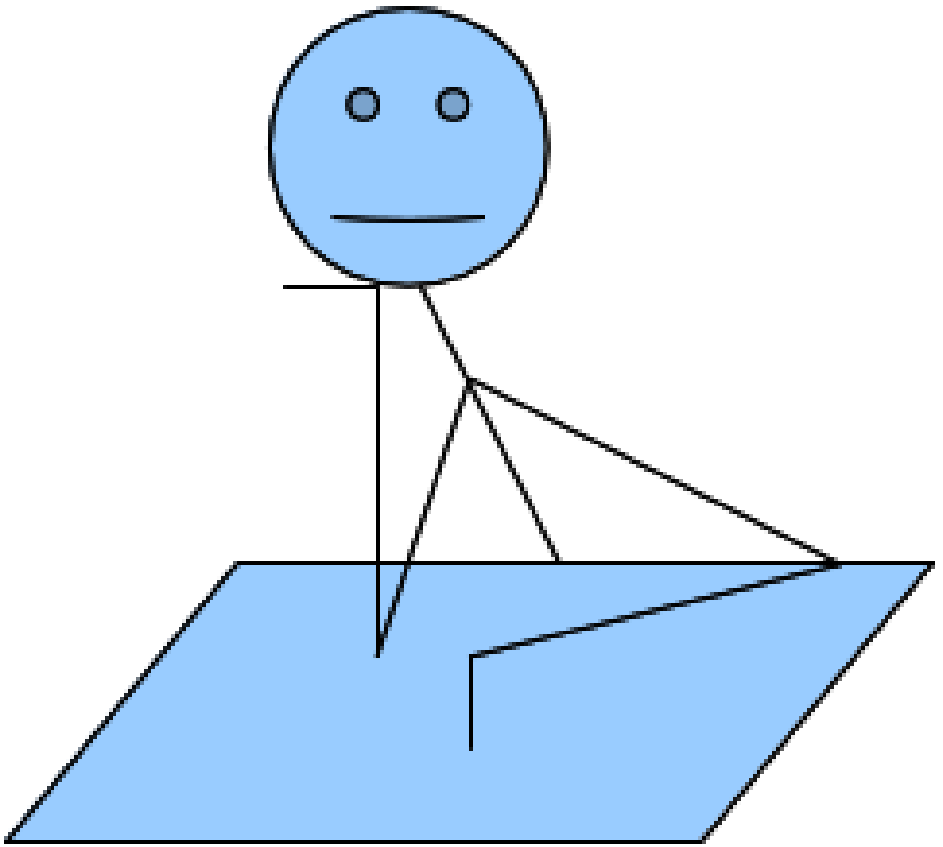
```
int main(int argc, char* argv[])
{
  CURL *curl;
  CURLcode res;

  if  (argc < 2)
  {

fprintf(stderr,"Usage:\tfromServerHeaderExaminer
<url>\n");
    exit(EXIT_FAILURE);
  }

  const char*   urlCPtr         = NULL;
  urlCPtr       = argv[1];

  struct MemoryStruct chunk;

  chunk.memory = malloc(1);  /* will be grown as
needed by the realloc above */
  chunk.size = 0;    /* no data at this point */

  curl_global_init(CURL_GLOBAL_ALL);
  curl = curl_easy_init();
  if(curl) {
    struct MemoryStruct headers;

    headers.memory = malloc(1);
    headers.size = 0;  // no data at this point

    curl_easy_setopt(curl, CURLOPT_URL, urlCPtr);
    /* send all data to this function  */
    curl_easy_setopt(curl,
          CURLOPT_WRITEFUNCTION,
          WriteMemoryCallback);
    /* we pass our 'chunk' struct to the
callback function */
    curl_easy_setopt(curl, CURLOPT_WRITEDATA,
(void *)&chunk);

    curl_easy_setopt(curl,
 CURLOPT_HEADERFUNCTION,
 WriteMemoryCallback);
    curl_easy_setopt(curl,
  CURLOPT_HEADERDATA, &headers);

    res = curl_easy_perform(curl);
    if(res != CURLE_OK) {
      fprintf(stderr,
        "Curl perform failed: %s\n",
        curl_easy_strerror(res));
      return 1;
    }

    free(chunk.memory);
    headers.memory[headers.size] = '\0';
    printf("Headers:\n%s",headers.memory);
    curl_easy_cleanup(curl);
  }
  curl_global_cleanup();
  return 0;
}
```

Your Turn!

Write a program to get the content-type!

A better way for common fields

- Question: "*Must we cycle through all fields to find a desired value?*"
- Answer "*For common fields, no!*"



Obtaining common fields (Java URLConnection)

- `public String getContentType ()`
  - Returns content type in MIME format (e.g. image/jpeg).
  - May also have charset too
  - Returns empty string if not available (no throwing Exceptions)
- `public int getContentLength()`
  - How many bytes of content there are
  - -1 if field not present, or too big (> 2,147,483,647)
  - If too big consider `public long getContentLengthLong()` // Java 7
- `public String getContentEncoding()`
  - Returns null if unencoded
  - Most common encoding is x-gzip
  - NOTE: Different from charset encoding

Obtaining common fields (Java URLConnection) cont'd

- `public long getDate ()`
  - When doc was sent (long telling milliseconds since 1970 Jan 1, GMT)
  - 0: "No date field"
  - Turn into a `Date` object with:
    - `Date documentSent = new Date(urlConnect.getDate());`
- `public long getExpiration()`
  - Expiration time in milliseconds since 1970 Jan 1, GMT
  - After that, kick out of cache and reload it
- `public long getLastModified ()`
  - Last modification time in milliseconds since 1970 Jan 1, GMT

Caches

- Remember semantics of GET
  - Side-effect free!
  - Idempontent!
  - Book-markable!
  - ***Cachable!***
- Why reload resource if can store locally?
  - A commonly accessed homepage
  - Images shared by may pages of website
  - Laggy network, slow server

Cache-control options

- New with HTTP 1.1
- max-age=(seconds)
  - How many seconds from now resource expires
- s-maxage=(seconds)
  - How many seconds from now resource in ***shared cache*** expires (private caches may keep for longer)
- public
  - Okay to cache authenicated responses
- private
  - Cache, but only for you
- no-cache
  - Cache if you want, but check ETag or Last-modified in header
- no-store
  - Do ***not*** cache

Caches in Java

- JVM has infrastructure to do caching for you, but you have to set it up
- Class MemoryCache
  - You give it a URI and the URLConnection for a successful connection
    - “put()” method
  - To request the URLConnection back, give it:
    - URI, request method (must be “GET”) and headers
    - “get()” method
- SimpleCacheRequest (?)
- SimpleCacheResponse
  - Holds URLConnection and headers
- CacheControl
  - Caching policy
- NOTE: Following slides have been debugged from Chapter 7 of Elliotte Rusty Harold “*Java Network Programming: 4th Ed.*”

CacheControl

```
import java.util.Date;
import java.util.Locale;

public class CacheControl
{
  private Date  maxAge      = null;
  private Date  sMaxAge     = null;
  private boolean    mustRevalidate  = false;
  private boolean    noCache     = false;
  private boolean    noStore   = false;
  private boolean    proxyRevalidate = false;
  private boolean    publicCache   = false;
  private boolean    privateCache  = false;

  public CacheControl (String s)
  {
    if  (s == null)
      return;   // Default policy

    String value = s.trim();

    String[] components = value.split(",");

    Date now = new Date();

    for  (String component : components)
    {
      try
      {
component = component.trim().toLowerCase(Locale.US);

if  ( component.startsWith("max-age=") )
{
  int secondsInTheFuture = Integer.parseInt(component.substring(8));
  maxAge = new Date(now.getTime() + 1000 * secondsInTheFuture);
System.out.println("CacheControl = max-age");
}
else
if  ( component.startsWith("s-maxage=") )
{
  int secondsInTheFuture = Integer.parseInt(component.substring(9));
  sMaxAge = new Date(now.getTime() + 1000 * secondsInTheFuture);
System.out.println("CacheControl = s-maxage");
}
else
if  ( component.equals("must-revalidate") )
{
  mustRevalidate    = true;
System.out.println("CacheControl = must-revalidate");
}
else
if  ( component.equals("proxy-revalidate") )
{
  proxyRevalidate   = true;
System.out.println("CacheControl = proxy-revalidate");
}
```

CacheControl

```
else
if  ( component.equals("no-cache") )
{
  noCache   = true;
System.out.println("CacheControl = no-cache");
}
else
if  ( component.equals("public") )
{
  publicCache = true;
System.out.println("CacheControl = public");
}
else
if  ( component.equals("private") )
{
  privateCache    = true;
System.out.println("CacheControl = private");
}
    }
    catch  (RuntimeException ex)
    {
      continue;
    }
   }
  }
```

```
  public Date getMaxAge ()
  {  return(maxAge); }

  public Date getSharedMaxAge ()
  {  return(sMaxAge); }

  public boolean mustRevalidate ()
  {  return(mustRevalidate); }

  public boolean proxyRevalidate ()
  {  return(proxyRevalidate); }

  public boolean noStore ()
  {  return(noStore); }

  public boolean noCache ()
  {  return(noCache); }

  public boolean publicCache ()
  {  return(publicCache); }

  public boolean privateCache ()
  {  return(privateCache); }

}
```

SimpleCacheRequest

```
import java.io.*;
import java.net.*;

public class SimpleCacheRequest extends CacheRequest
{
  private ByteArrayOutputStream out = new ByteArrayOutputStream();

  @Override
  public OutputStream getBody ()
    throws IOException
  { return out; }

  @Override
  public void abort ()
  { out.reset(); }

  public byte[] getData ()
  {
    if  (out.size() == 0)
      return(null);
    else
      return(out.toByteArray());
  }
}
```

SimpleCacheResponse

```
import java.io.*;
import java.net.*;
import java.util.*;

public class SimpleCacheResponse extends CacheResponse
{
  private final Map<String,List<String>> headers;
  private final SimpleCacheRequest request;
  private final Date expires;
  private final CacheControl control;

  public SimpleCacheResponse   (SimpleCacheRequest  request,
        URLConnection        uc,
 CacheControl    control
)
    throws IOException
  {
    this.request    = request;
    this.control = control;
    this.expires = new Date(uc.getExpiration());
    this.headers   =
Collections.unmodifiableMap(uc.getHeaderFields());
  }

  @Override
  public InputStream getBody ()
  { return new ByteArrayInputStream(request.getData()); }

  @Override
  public Map<String,List<String>> getHeaders    ()
    throws IOException
  { return(headers) ; }

  public CacheControl getControl ()
  { return(control); }

  public boolean isExpired ()
  {
    Date now = new Date();

    if  ( (control.getMaxAge() != null) &&
control.getMaxAge().before(now))
      return(true);

    if  ( (expires != null) && (control.getMaxAge() !=
null) )
      return(expires.before(now));

    return(false);
  }
}
```

MemoryCache

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.concurrent.*;

public class MemoryCache extends ResponseCache
{
  private final Map<URI,SimpleCacheResponse>   responses
        = new ConcurrentHashMap<URI,SimpleCacheResponse>();

  private final int maxEntries;

  public MemoryCache ()
  { this(100); }

  public MemoryCache (int maxEntries)
  { this.maxEntries = maxEntries; }

  @Override
  public CacheRequest put (URI uri, URLConnection conn)
    throws IOException
  {
    HttpURLConnection httpConn = ((HttpURLConnection)conn);

    System.out.println("Caching: for " + uri);
    System.out.println(httpConn.getRequestMethod());
    System.out.println();

    if  (responses.size() >= maxEntries)
      return(null);

    CacheControl control    =
new CacheControl(httpConn.getHeaderField("Cache-Control"));

    if  ( control.noStore() )
      return null;

    if  ( !httpConn.getRequestMethod().startsWith("GET") )
      return null;    //  Only cache GET

    SimpleCacheRequest    request    = new SimpleCacheRequest();
    SimpleCacheResponse response =new
SimpleCacheResponse(request,httpConn,control);

    responses.put(uri,response);
    return(request);
  }


  @Override
  public CacheResponse get (URI          uri,
                String          requestMethod,
 Map<String,List<String>> requestHeaders
)
    throws IOException
  {
    System.out.print("Looking in cache for " + uri + ": ");

    if  ("GET".equals(requestMethod))
    {
```

MemoryCache

```
      if  (response != null  && response.isExpired())
      {
        System.out.println("It is expired!");
        responses.remove(response);
        response = null;
      }

      if  (response == null)
      {
        System.out.println("Not found!");
      }
      else
      {
        System.out.println("Got it!");
      }

      return(response);
    }
    else
    {
      System.out.println("Not found.");
      return(null);
    }
  }

}
```

A Java testing program

```
import java.io.*;
import java.net.*;
import java.nio.charset.StandardCharsets;

public class CachingEncodingAwareSourceViewer
{
  public static void main (String[] args)
  {
    ResponseCache.setDefault(new
MemoryCache());
    while  (true)
    {
      String urlString;
      BufferedInputStream bf =
        new BufferedInputStream(System.in);
      BufferedReader stdInReader =
        new BufferedReader
        (new InputStreamReader
(bf, StandardCharsets.UTF_8)
);

      System.out.println("Please enter a URL
(or blank line to quit): ");
      try
      {
urlString = stdInReader.readLine();
      }
      catch  (IOException ex)
      {
        System.err.println(ex);
        break;
      }

      if  (urlString.equals("") )
        break;

      for (int i = 0;  i < 10;  i++)
        System.out.println();

      try
      {
String  encoding = "ISO-8859-1";
URL u  = new URL(urlString);
URLConnection
uc   = u.openConnection();
String  contentType
 = uc.getContentType();
int encodingStart
 = contentType.indexOf("charset=");

if  (encodingStart != -1)
{
  encoding  =
contentType.substring(encodingStart + 8);
}
```

A Java testing program, cont'd

```
InputStream in  = new BufferedInputStream(uc.getInputStream());
Reader  r = new InputStreamReader(in,encoding);

int c;

while  ((c = r.read()) != -1)
{
  System.out.print((char)c);
}

r.close();
      }
      catch  (MalformedURLException ex)
      {
        System.err.println(urlString + " is not a parseable URL");
      }
      catch  (UnsupportedEncodingException ex)
      {
System.err.println
("Server sent an encoding Java does not supported: " +
 ex.getMessage()
);
      }
      catch  (IOException ex)
      {
        System.err.println(ex);
      }
    }
  }
}
```

And the simplest of servers

```
// onePageCacheTestingHttpServer.c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>//For socket()
#include <netinet/in.h>//For sockaddr_in and htons()
#include <netdb.h> //For getaddrinfo()
#include <errno.h> //For errno var
#include <sys/stat.h> //For open(), read(),write()
#include <fcntl.h> // and close()
#include <signal.h>
#include <wait.h>
#include <time.h>
#include <unistd.h>   // open()

const int BUFFER_LEN    = 4096;
const int DEFAULT_PORT_NUM  = 20001;
const int MIN_PORT_NUM    = 1024;
const int MAX_PORT_NUM  = 65535;

#define   HEADER_TEMPLATE    \
"HTTP/1.1 200 OK\r\n"      \
"Server: JoesSillyAssServer/1.0\r\n"   \
"Content-Type: text/html\r\n"    \
"Content-Length: %d\r\n" \
"Last-Modified: %s"  \
"Cache-Control: %s\r\n"    \
"Date: %s"  \
"\r\n"
```

```
#define   RESOURCE_TEMPLATE   \
"<!DOCTYPE HTML>"       \
"<html lang=\"en\">" \
" <head><title>My only page</title></head>"  \
" <body>"   \
"  <h1>My only page</h1>"   \
"  Downloaded %s"    \
" </body>"  \
"</html>"

const char* CACHE_CONTROL[] =  {"max-age=",
 "s-maxage=",
 "public",
 "private",
 "no-cache",
 "no-store",
       };
void showUsage()
{
  fprintf(stderr,
  "Usage:\tonePageCacheTestingHttpServer <howCache> <port>\n"
  "Where:\n"
  "  <howCache> is:\n"
  "\tmax-age=(seconds)\n"
  "\ts-maxage=(seconds)\n"
  "\tpublic\n"
  "\tprivate\n"
  "\tno-cache\n"
  "\tno-store\n"
  "  <port> is integer in [%d..%d], default is %d\n",
  MIN_PORT_NUM,MAX_PORT_NUM,DEFAULT_PORT_NUM
 );
}
```

And the simplest of servers, cont'd

```
int    main       (int   argc, char* argv[])
{
  int  port = DEFAULT_PORT_NUM;
  int  cacheControlIndex;
  int  cacheControlSeconds;
  size_tcacheControlStrLen;
  char   cacheControlBuffer[BUFFER_LEN];

  if  (argc < 2)
  {
    showUsage();
    exit(EXIT_FAILURE);
  }

  switch  (argc)
  {
  default :
  case 3 :
    port = strtol(argv[2],NULL,0);

    //  Do *not* break, go on to handle argv[1]:

  case 2 :
    for  (cacheControlIndex = 0;
      cacheControlIndex < sizeof(CACHE_CONTROL)/sizeof(const
char*);
  cacheControlIndex++
 )
  {
    const char*   controlPtr   = CACHE_CONTROL[cacheControlIndex];

    cacheControlStrLen    = strlen(controlPtr);

    if  (strncmp(controlPtr,argv[1],cacheControlStrLen) == 0)
      break;
  }
 }

  if  ( (port < MIN_PORT_NUM) || (port > MAX_PORT_NUM) )
  {
    fprintf(stderr,"Illegal port number.\n\n");
    showUsage();
    exit(EXIT_FAILURE);
  }

  if  (cacheControlIndex >= sizeof(CACHE_CONTROL)/sizeof(const
char*))
  {
    fprintf(stderr,"Illegal cache option.\n\n");
    showUsage();
    exit(EXIT_FAILURE);
  }

  if  (argv[1][cacheControlStrLen-1] == '=')
  {
    char*   cPtr;

    cacheControlSeconds  = strtol(argv[1] +
cacheControlStrLen,&cPtr,0);
```

And the simplest of servers, cont'd

```
  if  ( (cPtr != '\0') || (cacheControlSeconds < 0) )
  {
    fprintf(stderr,"Illegal cache option.\n\n");
    showUsage();
    exit(EXIT_FAILURE);
  }

}

// Create a socket
int socketDescriptor = socket(AF_INET, // AF_INET domain
      SOCK_STREAM, // Reliable TCP
      0);

// We'll fill in this datastruct
struct sockaddr_in socketInfo;
// Fill socketInfo with 0's
memset(&socketInfo,'\0',sizeof(socketInfo));

// Use std TCP/IP
socketInfo.sin_family = AF_INET;

// Tell port in network endian with htons()
socketInfo.sin_port = htons(port);

// Allow machine to connect to this service
socketInfo.sin_addr.s_addr = INADDR_ANY;

// Try to bind socket with port and other specifications
int status = bind(socketDescriptor, // from socket()
  (struct sockaddr*)&socketInfo,
  sizeof(socketInfo)
 );
```

```
if  (status < 0)
{
  fprintf(stderr,"Could not bind to port %d\n",port);
  exit(EXIT_FAILURE);
}

listen(socketDescriptor,5);

char   header[BUFFER_LEN];
char   resource[BUFFER_LEN];
int  i;
time_t    current_time;
char* timeCPtr;
int  headerLen;
int  resourceLen;

printf("Please connect to http://127.0.0.1:%d\n",port);

for  (i = 0;  i < 4;  i++)
{
  // Accept connection to client
  int clientDescriptor = accept(socketDescriptor,NULL,NULL);

  current_time = time(NULL);

  if (current_time == ((time_t)-1))
  {
     (void) fprintf(stderr, "Failure to obtain the current time.\n");
     exit(EXIT_FAILURE);
  }
```

And the simplest of servers, cont'd

```
    timeCPtr  = asctime(gmtime(&current_time));
    resourceLen=
snprintf(resource,BUFFER_LEN,RESOURCE_TEMPLATE,timeCPtr);
    headerLen   = snprintf
(header,BUFFER_LEN,HEADER_TEMPLATE,
 resourceLen,timeCPtr,argv[1],timeCPtr
);
    write(clientDescriptor,header,headerLen);
    write(clientDescriptor,resource,resourceLen);
    write(STDOUT_FILENO,header,headerLen);
    close(clientDescriptor);
  }

  close(socketDescriptor);
  return(EXIT_SUCCESS);
}
```

And caching with libcurl?

Sorry, you will have to do that yourself
(Well, we pretty much did it ourselves
with all that Java! ***Whew!***)

Your turn!

Don’t be shy now!
Test the Java program to see if it caches!

Run server as:
$ ./onePageCacheTestingHttpServer public

Run client as:
$ java CachingEncodingAwareSourceViewer

Your turn again!

MemoryCache.java obviously
saves the cache in memory.

Where else/how else could it save the cache?

Configuring the Connection (Java)

- These methods govern how the URLConnection works:
- public URL getURL()
  - Returns URL (obviously)

Configuring the Connection (Java)

- These methods govern how the URLConnection works:
  - public void setAllowUserInteraction(boolean allow)
  - public boolean getAllowUserInteraction()
- By default user interaction is **not** allowed
- Have to turn on if want to:
  - Login with username and password

Configuring the Connection (Java)

- These methods govern how the URLConnection works:
  - public void setDoInput(boolean doInput)
  - public boolean getDoInput()
- By default **is** allowed
- Must be on if want to:
  - Read from server

Configuring the Connection (Java)

- These methods govern how the URLConnection works:
  - public void setDoOutput(boolean doOutput)
  - public boolean getDoOutput()
- By default is **not** allowed
- Must be on if want to:
  - Write to server

Configuring the Connection (Java)

- These methods govern how the URLConnection works:
  - public void setUseCaches(boolean useCaches)
  - public boolean getUseCaches()
- Allow the usage of local cache (if available)
- By default **is** allowed
- Must be off if want to:
  - Force Java to get most recent file from server

Configuring the Connection (Java)

- These methods govern how the URLConnection works:
  - public void setIfModifiedSince(boolean doInput)
  - public long getIfModifiedSince()
- Tells Java to tell server to send page only if modified since the specified date
- Server can respond with either:
  - HTTP/1.0 304 Not Modified
  - Or just returning it
- Time in milliseconds since midnight GMT, 1970 Jan 1

Configuring the Connection (Java)

- These methods govern how the URLConnection works:
  - public void setConnectTimeout(int timeout)
  - public int getConnectTimeout()
  - public void setReadTimeout(int timeout)
  - public int getReadTimeout()
- setConnectTimeout()/getConnectTimeout()
  - For waiting to establish initial connection
- setReadTimeout()/getReadTimeout()
  - For waiting to read data
- Time in milliseconds
- 0 means “never timeout”
  - Negative numbers throw IllegalArgumentException

Changing the client to server header

- These methods govern how the URLConnection works:
  - public void setRequestProperty(String name, String value)
  - e.g. uc.setRequestProperty("cookie","user=itIsMe");
  - public void addRequestProperty(String name, String value)
- setRequestProperty()
  - For existing properties
- addRequestProperty()
  - For new properties

HTTP methods other than GET

- Open a connection
- Specify doing output (by default will use POST)
- Make and use OutputStreamWriter from getOutputStream() of URLConnection:

```
URLConnection uc = url.openConnection();
uc.setDoOutput(true);
try  (OutputStreamWriter out =
          new OutputStreamWriter(uc.getOutputStream(),"UTF-8")
      )
{
  out.write(stringToSend + "\r\n");
  out.flush();
}
```

QueryString.java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;

public class QueryString
{
  private StringBuilder query = new
StringBuilder();

  public QueryString()
  { }

  public synchronized void add (String name,
String value)
  {
    if  (query.length() > 0)
      query.append('&');

    encode(name,value);
  }

  private synchronized void encode (String name, String value)
  {
    try
    {
      query.append(URLEncoder.encode(name,"UTF-8"));
      query.append('=');
      query.append(URLEncoder.encode(value,"UTF-8"));
    }
    catch  (UnsupportedEncodingException ex)
    {
      throw new RuntimeException("Broken VM does not support
UTF-8");
    }
  }

  public synchronized String getQuery ()
  {
    return(query.toString());
  }

  @Override
  public String toString ()
  {
    return(getQuery());
  }
}
```

FormPoster.java

```
import  java.io.*;
import  java.net.*;

public class    FormPoster
{
  private URL   url;
  private QueryString   query = new QueryString();

  public FormPoster (URL url)
  {
    if (!url.getProtocol().toLowerCase().startsWith("http"))
      throw new IllegalArgumentException("Posting only works with http");

    this.url = url;
  }

  public void add (String name, String value)
  {
    query.add(name,value);
  }

  public URL getURL()
  {
    return(this.url);
  }

  public InputStream post () throws IOException
  {
    // open the connection and prepare it to POST:
    URLConnection uc = url.openConnection();
    uc.setDoOutput(true);
    try  (OutputStreamWriter out =
          new OutputStreamWriter(uc.getOutputStream(),"UTF-8")
         )
    {
      out.write(query.toString());
      out.write("\r\n");
      out.flush();
    }

    // Return the response
    return uc.getInputStream();
  }


  public static void main (String[] args)
  {
    URL url;
    if  (args.length > 0)
    {
      try
      {
        url = new URL(args[0]);
      }
      catch (MalformedURLException ex)
      {
        System.err.println("Usage: java FormPoster <url>");
        System.err.println("  e.g. java FormPoster http://127.0.0.1:20001");
        return;
      }
    }
```

FormPoster.java, cont’d

```
    else
    {
      System.err.println("Usage: java FormPoster <url>");
      System.err.println("  e.g. java FormPoster http://127.0.0.1:20001");
      return;
    }

    FormPoster poster = new FormPoster(url);
    poster.add("name","Joe Phillips");
    poster.add("email","jphillips@cdm.depaul.edu");

    try  (InputStream in = poster.post())
    {
      // Read the response
      Reader r = new InputStreamReader(in);
      int c;

      while  ((c = r.read()) != -1)
      {
        System.out.print((char)c);
      }

      System.out.println();
    }
    catch  (IOException ex)
    {
      System.err.println(ex);
    }
  }
}
```

HTTP methods other than get

- A fancier example, explicitly give HTTP method (by “mkyong”)

```
HttpsURLConnection con = (HttpsURLConnection) url.openConnection();
//add reuqest header
con.setRequestMethod("POST");
con.setRequestProperty("User-Agent", USER_AGENT);
con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
String urlParameters =
"sn=C02G8416DRJM&cn=&locale=&caller=&num=12345";

// Send post request
con.setDoOutput(true);
DataOutputStream wr =
    new DataOutputStream(con.getOutputStream());
wr.writeBytes(urlParameters);
```

References:

- Elliotte Rusty Harold “*Java Network Programming: 4th Ed.*”
- mkyong https://www.mkyong.com/java/how-to-send-http-request-getpost-in-java/ May 25, 2013