RQs + PEs: Stream-based Implementation of Application Protocols

Advanced Topics in Java

Khalid Azim Mughal
khalid@ii.uib.no
http://www.ii.uib.no/~khalid/atij/

Version date: 2006-09-04

REVIEW QUESTIONS - Application Protocols

1. Explain what is the TCP/IP stack.
2. What is HTTP (HyperText Transmission Protocol) primarily used for? How do a client and a server utilize the protocol?
3. Explain the terms URI, URL and URN. Name the common components of a URL.
4. What is a cookie and what is it's purpose?
5. What is CGI and what is it's purpose?
6. What are the main components of a HTTP message?
7. Explain the format of a HTTP request, distinguishing between GET, HEAD and POST methods.
8. Name three HTTP request headers and their purpose.
9. Explain the format of a HTTP response, distinguishing between it's main components.
10. Explain why HTTP is a line-based, stateless protocol.
11. Name three HTTP response header fields and their purpose.
12. Name three HTTP entity header fields and their purpose.
13. Sketch the socket-based communication between a client and a server. Pay attention

to how they use input and output streams to communicate.

14. What is the purpose of encoding and decoding URLs?

PROGRAMMING EXERCISES - Application Protocols

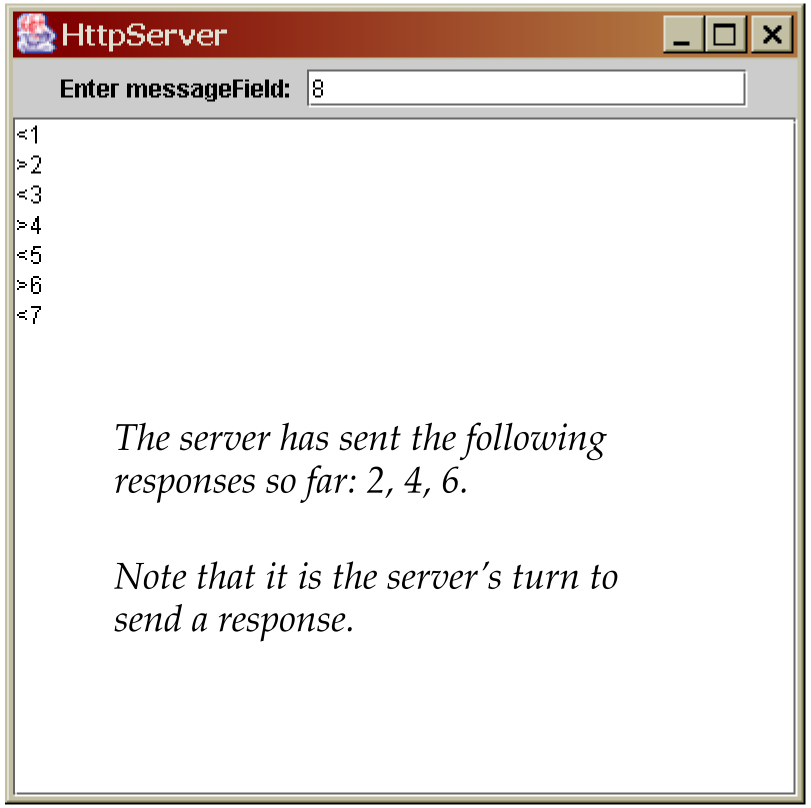
1. We will implement a chat program where a client and a server communicate with each other by passing HTTP messages to each other. The client sends a HTTP request and then waits for a HTTP response. The server sends a HTTP response and then waits for a new HTTP request. The communication is strictly HTTP request/ response-based.

   Write a client (`HttpClient`) and a server (`HttpServer`) that transmit and receive lines of text packaged in HTTP messages. Both have a simple GUI to enter a line of text. For example, when the client enters a line by hitting RETURN, the line of text is sent to the server in a HTTP request. The client then waits for the server to send a HTTP response.

   However, it is the client that initiates the communication by sending a HTTP request to the server. It should be possible for either of them to terminate the communication by entering an empty line in the GUI.

   The GUI is continuously updated to show the dialog between the client and the server. See illustration of client GUI and server GUI interaction on the next page.

   You can modify the source code files (`NewTCPclient.java`, `NewTCPserver.java`, `NewTCPgui.java`) to create a chat program which uses HTTP messages.

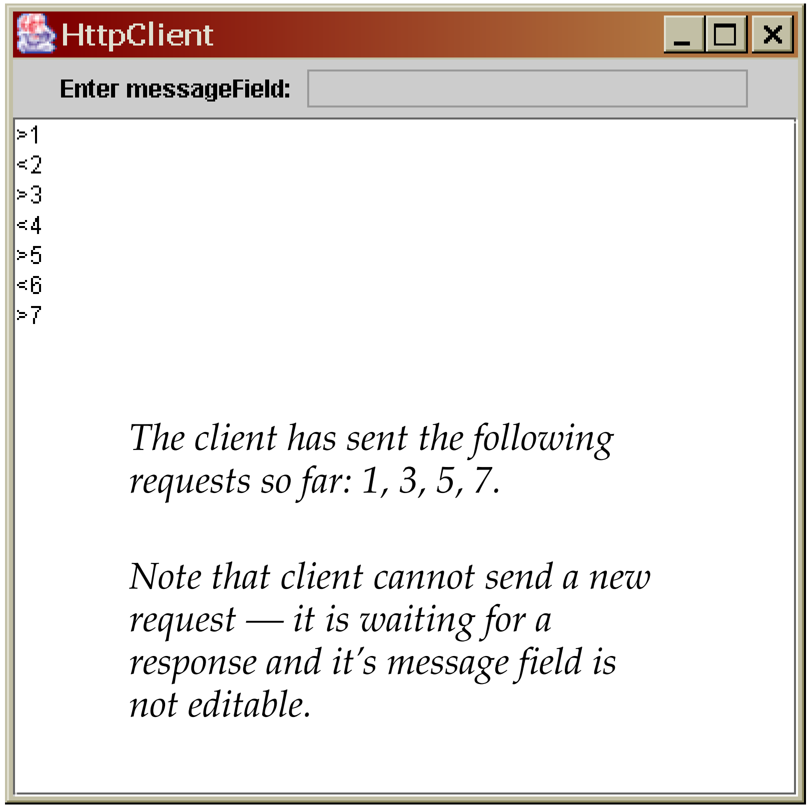


Enter messageField: 8

```
<1
>2
<3
>4
<5
>6
<7
```

The server has sent the following responses so far: 2, 4, 6.

Note that it is the server's turn to send a response.



Enter messageField:

```
>1
<2
>3
<4
>5
<6
>7
```

The client has sent the following requests so far: 1, 3, 5, 7.

Note that client cannot send a new request — it is waiting for a response and it's message field is not editable.