Internetprogrammierung

http://proglang.informatik.uni-freiburg.de/teaching/inetprog/2006/

Übungsblatt 1

10.5.2006

Aufgabe 1 (nslookup)
In der Vorlesung wurde nslookup als textuelles Werkzeug für DNS-Anfragen vorgestellt. Hangeln sie sich mit Hilfe von nslookup von der Stammdomäne bis hinunter zur untersten Subdomaine, um eine autoritative IP-Adresse von www.abc.com.au zu erhalten. Gehen sie dabei vor wie bei einer iterativen DNS-Anfrage. Damit nslookup die Adresse nicht selbst auflöst, müssen sie zu Beginn der Sitzung die Befehle set norec und set nosearch eingeben. Mit dem Befehl server *ip-address* wechseln sie den Domainserver den nslookup benutzt um an Informationen zu gelangen. Geben sie für diese Aufgabe eine (sinnvoll gekürzte) Aufzeichnung ihrer Sitzung mit nslookup ab.

Aufgabe 2 (Mails verschicken mit Java, Teil 1)
Schreiben sie ein Java Programm, welches Emails über einen SMTP-Server versendet. Der Host und Port des SMTP-Servers, die Absender- und Empfängeradresse, der Betreff sowie der Text der Email sollen als Kommandozeilenargumente übergeben werden. Das Verschicken der Email soll in einer Methode sendMail der Klasse SMTPServer erfolgen. Diese Methode hat die Signatur public void sendMail(String from, String to, String subject, String text) throws Exception.

Zum Verschicken der Email sollen sie eine Verbindung zu einem SMTP-Server aufbauen. Von einem Rechner der Informatik Fakultät können sie etwa den Host smtp an Port 25 benutzen. (Achtung: von außerhalb der Informatik ist Zugang zum SMTP-Server nur über SSL möglich.) SMTP (Simple Mail Transfer Protocol) ist in RFC 821 (http://www.ietf.org/rfc/rfc0821.txt) definiert. Die nachfolgende Tabelle listet die für sie relevanten SMTP Befehle auf. SMTP Befehle werden durch die Zeichenfolge <CR><LF> beendet. Dabei steht <CR> für "Carriage Return" (ASCII-Code 13) und <LF> für "Line Feed" (ASCII-Code 10).

HELO *domain*	Identifiziert den Client.
MAIL FROM: *address*	Initiiert das Verschicken einer Email und setzt den Absender.
RCPT TO: *address*	Spezifiziert den Empfänger.
DATA	Nach diesem Befehl erwartet der SMTP-Server den Text der Email. Dabei müssen auch Header wie From, To oder Subject angegeben werden. Ein Punkt am Anfang einer Zeile direkt gefolgt von <CR><LF> beendet den Text der Email.
QUIT	Beendet die Verbindung.

Antworten des SMTP-Servers beginnen immer mit einem dreistelligen Statuscode. Nachfolgende Tabelle listet die wichtigsten Codes auf:

220	Der Server ist bereit.
250	Der vorangehende Befehl war in Ordnung.
354	Der Server ist bereit den Text der Email zu empfangen.
221	Der Server schließt die Verbindung.

Bevor sie mit der Programmierung beginnen, sollten sie telnet benutzen, um eine SMTP-Sitzung durchzuspielen. Aufruf (funktioniert nur innerhalb der Informatik): `telnet smtp 25`.

Aufgabe 3 (Mails verschicken mit Java, Teil 2)
Die vorangehende Aufgabe zeigt, dass das SMTP-Protokoll nicht sonderlich gut für die Benutzung durch Menschen geeignet ist. In dieser Aufgabe soll nun ein eigener kleiner Mailserver geschrieben werden, der ein auf menschliche Benutzer ausgerichtetes, selbstdefiniertes Protokoll versteht.

Das Protokoll ist sehr einfach gehalten: Sobald sich ein Client (z.B. via Telnet) mit dem Mailserver verbindet, fragt der Server den Client, ob er eine Email-Nachricht verschicken möchte. Antwortet der Client mit `yes`, fragt der Server nacheinander nach Sender- und Empfängeradresse, sowie nach Betreff und Inhalt der Email. (Sie können sich beim Inhalt auf eine Zeile beschränken.) Dann wird die Email verschickt und der Mailserver fragt den Client ob eine weiter Email verschickt werden soll. Ist die Antwort `yes` geht die Sache von vorne los. Hier ist eine Beispielsitzung (Benutzereingaben sind kursiv gedruckt):

Das Programm soll als eine Java Klasse `MailServer` implementiert werden. Der Host und der Port des SMTP-Servers, welcher zum eigentlichen Versenden der Emails benutzt wird, steht dem Programm als Kommandozeilenargument zur Verfügung, ebenso wie der Port, auf dem der Mailserver auf Anfragen wartet. Zum Versenden der Emails wird natürlich die Klasse `SMTPServer` aus der vorhergehenden Aufgabe benutzt. Falls sie diese Aufgabe nicht gelöst haben, können sie die z.B. die *Commons Email* Bibliothek aus dem Jakarta Project (`http://jakarta.apache.org/commons/email/`) benutzen.

Geben sie für diese Aufgabe (zusätzlich zum Code) die Aufzeichnung einer Sitzung mit ihrem Mailserver ab.

Zusatzfrage (fließt nicht in die Bewertung ein): Können sich mehrere Clients gleichzeitig mit ihrem Mailserver verbinden? Falls nicht, wie könnte man dieses Feature implementieren?

Abgabe: 17.5.2006
Die Abgabe erfolgt in der Übungsstunde. Code soll in gedruckter Form abgegeben werden. Jede Aufgabe wird mit vier Punkten bewertet. Für Plagiate werden keine Punkte vergeben.