CS 580 Client-Server Programming
Spring Semester, 2004
Doc 21 Some on Protocol

Contents

Doc 21 Some on Protocol ................................................................. 1
Some low level Parsing ................................................................. 2
Calendar System ........................................................................... 8

References

Java Network Programming, Harold, O’Reilly, pp 75-116
VisualWorks Internet Client Developer’ Guide, pp 24-48

Copyright ©, All rights reserved. 2004 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.
Some low level Parsing
Java String methods

"cat;man;ran".split(";");

Returns an array of String { “cat”, “man”, “ran”};

See http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html#sum
for valid arguments of split().

**StringTokenizer**

parts = new java.util.StringTokenizer("cat,man;ran;,fan", ",;"); while (parts.hasMoreElements())
{
    System.out.println( parts.nextToken());
}

**Output**
cat
man
ran
fan
Some Useful Smalltalk Collection Methods

'cat;man;ran' tokensBasedOn: $;

Result

OrderedCollection ('cat' 'man' 'ran')

'cat. man... ran.'

piecesCutWhere:

[:each :next | each = $. and: [next = Character space]]
do: [:each | Transcript show: each printString; cr]

Result

'cat.'
' man...'
' ran.'
Some Useful Smalltalk Collection Methods

'cat\man\ran'
  runsFailing: [:each | each = $]
  do: [:each | Transcript show: each; cr]

Result

cat
man
ran

'cat\man\ran'
  runsSatisfying: [:each | each ~= $]
  do: [:each | Transcript show: each; cr]

Result

cat
man
ran
Java Streams

Java Streams do not have many methods that aid in parsing.

Avoid PrintStream – println() is platform dependent.

"PrintStream is evil and network programmers should avoid it like the plague."

readLine()

Text claims that readLine() is buggy.

Avoid using this method to read data from a socket.

Data Input/Output Streams

Are used for binary data.

Don’t use unless protocol is binary.

If protocol is binary these streams are only good between Java clients and servers.
Smalltalk Streams – Some Useful Methods

peek
Answer what would be returned with a self next, without changing position. If the receiver is at the end, answer nil.

peekFor: anObject
Answer false and do not move the position, if the next object is not anObject, or if the receiver is at the end. Answer true and increment the position if the next object is anObject.

skipToAll: aCollection
Skip forward to the next occurrence (if any) of aCollection. If found, leave the stream positioned before the occurrence, and answer the receiver; if not found, answer nil, and leave the stream positioned at the end.

throughAll: aCollection
Answer a subcollection from the current position through the occurrence (if any, inclusive) of aCollection, and leave the stream positioned after the occurrence. If no occurrence is found, answer the entire remaining stream contents, and leave the stream positioned at the end.

upTo: anObject
Answer a subcollection from position to the occurrence (if any, exclusive) of anObject. The stream is left positioned after anObject. If anObject is not found answer everything.

upToAll: aCollection
Answer a subcollection from the current position up to the occurrence (if any, not inclusive) of aCollection, and leave the stream positioned before the occurrence. If no occurrence is found, answer the entire remaining stream contents, and leave the stream positioned at the end.

skipUpTo: anObject
Skip forward to the occurrence (if any, not inclusive) of anObject. If not there, answer nil. Leaves positioned before anObject.
upToAll: and Java

upToAll: is a useful method

sdsu.io.ChunkReader

http://www.eli.sdsu.edu/java-SDSU/docs/sdsu/io/ChunkReader.html

Reads up to a given string in a stream or string

read = new sdsu.io.ChunkReader("catEOMmatEOM", "EOM")
while (read.hasMoreElements() )
{
    System.out.println( read.readChunk());
}

Result

cat
mat
Calendar System

Anonymous Users

• View the items in a calendar
• Do searches for particular events

Search on

• Date
• Title
• Location

Registered Users

After login on with user name and password can

• Enter new events
• Modify event they submitted earlier
• Cancel events they submitted earlier

Event

An event has

• A Title
• A start time (day and time)
• A duration, which could be all day
• A short description
• A location, which could be a room in a building, a building or a general location like Aztec Green
• The name of a submitter
Partial Sample Protocol

Request General Format

requestName CRLF
key1=value1 CRLF
key2=value2 CRLF
EOM CRLF

Sample

login CRLF
user=whitney CRLF
password=foobar CRLF
EOM CRLF

newEvent CRLF
title=The Emerging Cyberinfrastructure CRLF
start=April 2, 2004 11:00 CRLF
duration=60 CRLF
location=GMCS 327 CRLF
owner=whitney CRLF
EOM CRLF

Some issues are not specified on purpose
How to Handle Protocol

CalendarLogonRequest

Represents a Logon request  
Can generate valid Calendar logon protocol request string  
Can read valid Calendar logon protocol request
Java Example

public class CalendarLogonRequest {
    private Hashtable data = new Hashtable();

    public void userName( String name) { data.put("user", name); }

    public String userName() { return data.get("user"); }

    public void password(String password) {
        data.put("password", password);
    }

    public String toString() {
        StringBuffer logon = new StringBuffer();
        logon.append("login").
        logon.append(\r\n).
        logon.append("user=").
        logon.append( data.get("user") ).
        logon.append("\r\n").
        logon.append("password=").
        logon.append( data.get("password") ).
        logon.append("\r\n").
        logon.append("EOM\r\n").
        return logon.toString();
    }

    public static CalendarLogonRequest from(InputStream in) {
        parse in
        return CalendarLogonRequest object with username and password set
    }
}
Smalltalk Example

Smalltalk defineClass: #CalendarLogonRequest
    superclass: #{Core.Object}
    indexedType: none
    private: false
    instanceVariableNames: 'userName password '
    classInstanceVariableNames: "
    imports: "
    category: 'Protocol Examples'

CalendarLogonRequest class methods

from: aStringOrReadStream
    ^super new from: aStringOrReadStream readStream

userName: aNameString password: aPasswordString
    ^super new setUserName: aNameString password: aPasswordString

CalendarLogonRequest Instance methods

setUserName: aNameString password: aPasswordString
    userName := aNameString.
    password := aPasswordString

from: aReadStream
    | message lines |
    message := aReadStream upToAll: 'EOM\' withCRs.
    lines := message tokensBasedOn: Character cr.
    lines do:
        [:each | (each includes: $=) ifTrue:[self addPairFrom: each]]
addPairFrom: aString
    | keyValue |
    keyValue := aString tokensBasedOn: $=.
    keyValue first asLowercase = 'password'
        ifTrue: [ password := keyValue last ].
    keyValue first asLowercase = 'user'
        ifTrue: [ userName := keyValue last ].

asString
    | aStream |
    aStream := WriteStream on: (String new: 16).
    self printOn: aStream.
    ^aStream contents

printOn: aStream
    aStream
        nextPutAll: 'login'; cr;
        nextPutAll: 'user=';
        nextPutAll: userName; cr;
        nextPutAll: 'password=';
        nextPutAll: password; cr;
        nextPutAll: 'EOM'; cr

password
    ^password

terName
    ^userName
What is the Point?

Use CalendarLogonRequest to generate logon protocol

logonRequest = new CalendarLogonRequest();
logonRequest.user(“whitney”);
logonRequest.password(“foo”);

clientSocket = new Socket(“127.0.0.1”, 9009);
toServer = clientSocket.getOutputStream();

request = logonRequest.toString();
toServer.write(request.getBytes());

Use CalendarLogonRequest to read protocol on server side
Sample Client

clientSocket := SocketAccessor newTCPclientToHost: '127.0.0.1' port: 9009.
logon := CalendarLogonRequest userName: 'roger' password: 'foo'.
toServer := clientSocket readAppendStream.
toServer lineEndCRLF.
toServer nextPutAll: logon asString;flush.

Sample Server

server := SocketAccessor newTCPserverAtPort: 9009.
server listenFor: 5.

acceptedSocket := server accept.
[ | stream |
  stream := acceptedSocket readAppendStream.
  stream lineEndCRLF.
  request := CalendarLogonRequest from: stream.
  stream nextPutAll: 'Done'; commit.
  stream close.
] forkAt: Processor userSchedulingPriority -1.
Second Idea – Command

One the server side let the request actually perform the operation

public class CalendarLogonRequest {

    public Result execute(CalendarHandler aCalender) {
        check to see if user name and password are correct
        return a result object
    }
}
Third Idea – Protocol Stream

Create a CalendarInputStream

• Read returns a Calendar request object

```java
public void run(int port) throws IOException {
    ServerSocket input = new ServerSocket(port);
    log.info("Server running on port " + input.getLocalPort());
    while (true) {
        Socket client = input.accept();
        log.info("Request from " + client.getInetAddress());
        processRequest(  
            new CalendarInputStream(client.getInputStream()),
            new CalendarOutputStream(client.getOutputStream());
        client.close();
    }
}

doctor processRequest(  
    CalendarInputStream in,  
    CalendarOutputStream out)  
throws IOException {
    CalendarRequest request = (CalendarRequest) in.read();
    CalendarRequest response = request.execute(calendar);
    out.write(response);
}
```
**Issue – Which Request?**

Socket client = input.accept();

in = client.getInputStream();

String firstLine = in.readLine();
if (firstLine = “login”)  
    request = new CalendarLoginRequest();
else if (firstLine = “newEvent”)  
    request = new CalendarNewEventRequest();
else if etc.

**Can Use Prototype**

requests = new Hashtable();
requests.put(“login”, new CalendarLoginRequest());
requests.put(“newEvent”, new CalendarNewEventRequest());

Socket client = input.accept();

in = client.getInputStream();

String firstLine = in.readLine();
request = requests.get(firstLine).clone();