CS 580 Client-Server Programming
Fall Semester, 2002
Doc 17 Configuration Files
Contents

References

SDSU Java Library, http://www.eli.sdsu.edu/java-SDSU/docs/

VisualWorks Application Developers Guide, pages 219-223

CommanlLineInterest class>>example

Application Parameters & Configuration Files

Applications normal have configuration files to store

- User preferences
- Cached values
- Window settings
- Port numbers
- Database connection information
- Log file information
- Recent documents/web pages
- Cookies
- Values that need changing without recompiling

Some programs use environment variables

```
cvs co assignment2
```

Command line program/servers/utilities have flags

```
ls -la
ps -aux
```

Servers

Servers normally use configuration files & command line flags

Environment variables are uses much in servers (why?)

- Flag values override configuration file values
- Configuration file values override defaults in the code

Java

Some systems have libraries to handle config files & command line arguments

JDK does not seem to have such classes

There should be a number of Java libraries that provide such support

sdsu Java library is one such library

sdsu.util.ProgramProperties

Parses
- Configuration files
- Command line arguments

Command Line argument

- -flag=value
- -flag value
- -flag
- --xyz
- -- (ignore rest of the command line)

File Formats
- properties format

```
#A comment to the end of the line
key1=value1
key2=value2     with spaces
key3 with spaces=value3 #part of the value
```

- sdsu.util.LabeledData format

```
#A comment to the end of the line,
key1 = value1;
key2='value2    with spaces';
'key3 with spaces'=value3;              # a comment
```

Simple Example

```
import sdsu.util.ProgramProperties;

public class ConfigurationExample {

   public static void main(String args[]) {
      try
         {
         ProgramProperties flags =
            new ProgramProperties( args, "configurationFile");
         String nameValue =
            flags.getString( "name" , "No name given");
         int size = flags.getInt( "size", 0);
         boolean switchOn = flags.containsKey( "s");
         System.out.println( " nameValue: " + nameValue);
         System.out.println( " size: " + size);
         System.out.println( " switchOn: " + switchOn);

         }
      catch (java.io.IOException readParseProblem)
         {
         System.err.println( "Program aborted on error " +
         readParseProblem);
         }
   }
}
```

File "configurationFile.labeledData"

```
name=Roger;
size=12;
```

Sample Runs

java ConfigurationExample

Output

nameValue: Roger
size: 12
switchOn: false

java ConfigurationExample -s -name Pete

Output

nameValue: Pete
size: 12
switchOn: true

java ConfigurationExample -conf=otherFile

Output

nameValue: Sam
size: 8
switchOn: true

VisualWorks

Command Line Arguments

CEnvironment

Cenvironment class>>commandLine

Returns array of String of flags in order they were specified

Starting VisualWorks with:

```
visual script.im -port 5 -host=rohan.sdsu.edu -xyz
```

CEnvironment commandLine returns

```
#('visual' 'script.im' '-port' '5' '-host=rohan.sdsu.edu' '-xyz')
```

CEnvironment class>> userEnvironment

Returns CEnvironment object with user's environment variables

```
| user |
user :=CEnvironment userEnvironment.
user at: 'CVSROOT'
```

returns value of CVSROOT environment variable

CEnvironment is a subclass of Dictionary

Registering for Flags

To avoid parsing the command line array

Classes can register to be informed of individual flags

To register add method to CommanlLineInterest class
In dependencies-pragma protocol

Example

```
CommanlLineInterest class>>port: aTokenReadStream
   <triggerAtSystemEvent: #returnFromSnapshot option: '-port'>

   | port |
   port := aTokenReadStream next asNumber.
   SimpleDateServer port: port.

   "Now make SimpleDateServer>>port: set the port.
```

aTokenReadStream next returns the value after the flag

CommanlLineInterest class>>port: is called on startup when flag -port exits on command line

Only supports -flag value options

Can have method triggered at
- #earlySystemInstallation
- #returnFromSnapshot

#earlySystemInstallation flags are handled from left to right

Then #returnFromSnapshot flags are handled from left to right

Configuration Files

VisualWorks servers use configuration files

No system wide classes for supporting them

Too easy to create own system?