

CURL: Client for URL

- similar to FTP → sending/receiving files
- advantage → supports several protocols
 - FTP
 - SSH
 - SFTP
 - etc. etc.

```
curl -o ftp.xxx.yyy.file.iso # downloads some file
```

```
# THIS DOWNLOADS 3 FILES:
```

```
curl -OO ftp.xxx.yyy.disc{1,2,3}.iso # THIS IS NICE
```

```
curl -o ftp://John:fatcat@ftp.xxx.yyy/home/z.gtg -Q '-DELE z.gtg'
```

```
# THE ABOVE LOGS ON WITH USER NAME JOHN AND  
# PASSWORD fatcat and THEN GETS THE FILE  
# AND THEN DELETES THE SOURCE FILE
```

```
# IF YOU WANT TO UPLOAD:
```

```
curl -T instal.log ftp://John:fatcat@ftp.xxx.yyy.com/tmp/ -Q ↵  
"-RNFR instal.log" -Q "RNTO xinstal.log"
```

```
# ALSO, ABOVE CODE RENAMES LOCAL FILE instal.log  
# TO xinstal.log ON REMOTE SYSTEM
```

```
# LISTS ALL FILES IN 'BOOKS' DIRECTORY ON  
# THE REMOTE SYSTEM:
```

```
curl ftp://ftp.xxx.org/pub/loc/books/
```

FROM SOFTWARE ENGINEERING STANDPOINT,
THIS IS A BAD PROGRAM → IT'S HARD TO
UNDERSTAND & HARD TO USE. HMMMMMMM

WAR : Web Archive

- Made by Sun Microsystems using jar technology
- jar → Java Archiver → similar to tar syntax:

```
tar -cvf my.tar /tmp
```

```
jar -cvf my.jar /tmp
```

- * EVENTHOUGH ITEMS ARE IN JAR, YOU DO NOT NEED TO UN-JAR, YOU CAN JUST USE

```
jar -tvf my.jar # displays all files
```

- * WAR FILES ARE SPECIFIC FOR WEB APPLICATIONS!

/WEB-INF # you need to create this in a jar FILE!!!

SO, FIRST CREATE jar file, then mkdir

NOW MAKE A FILE:

web.xml # this defines structure of web application

ABBAS ASIDE:

```
tar -jcvf # using j utilizes BZ2 technology; VERY NICE!
```

IN ANY WEB APP, THERE'S CLIENT SIDE & SERVER SIDE.

- javascript collects data on client side
- then sends over web to server
- the server then returns info back & displays
- perl was old school of communicating between web server & database
- then PHP comes along; it's buddy buddy w/ MySQL
- then python → great language

- And, of course, we have RUBY!

* RUBY HAS ITS OWN MECHANISM, RAILS!!!

* Java was not part of this list (perl, php, py, ruby)
b/c Apache does not support it.

- Applet running on client-side

- servlet : technology allowing to run on server-side | sun's response

* Then agreement between Jakarta group, Jukes in charge of APACHE, makes agreement w/ JAVA Jukes but this is complicated b/c of JVM, so they make a new server & call it TOMCAT

* AND ON THE FRONT END : JSP → embedding Java in HTML
AND IT SENDS DATA TO TOMCAT

EAR: Enterprise Archive

- special enterprise format for Java Enterprise Edition
- packaging one or more modules into single archive so that deployment onto server happens simultaneously and coherently

Languages written on top of JAVA:

• groovy
• scala } both publicized in 2003 (groovy - us
scala - Switzerland)

• Martin Odersky, old sun dude, invented Scala

* SCALA is very powerful & very nice

→ it creates byte code that runs on JVM

- SCALA → like Ruby, very powerful
- Groovy → also simplifies JAVA, but Abbas doesn't like
- Scala: in Italian, it means stairs, but also a very famous opera house in MILAN, but has nothing to do with this, Abbas just wanted to let us know, Scala, rather, means scalability

TO RUN INTERACTIVELY:

```
$ scala
```

```
scala> # Now LET'S PARTY
```

IF you use 'val' it's constant; 'var' it's variable

```
scala> val msg: String = "Hello"
```

└──┬──
optional

```
scala> println(msg)
```

```
scala> msg = "bye" # throws error b/c msg is CONSTANT
```

COMMENTS ARE SAME AS C++ & JAVA // * /* */

```
scala> var age = 78
```

```
scala> println("I am " + age + " years old")
```

IN SCALA, EXACTLY LIKE RUBY, EVERYTHING'S OBJECT

```
def max(x: Int, y: Int): Int = return value if (x < y) y else x
```

```
println("largest of 5 and 9 is " + max(5, 9))
```

IF YOU WANT YOUR FUNCTION TO SPAN MANY LINES.

YOU CAN USE BRACES

```
def greet() = println("Hello world")
```

```
greet()
```

To pass args on the command line:

args(0) # arrays in scala use ~~br~~ parens not brackets

scala does not have shebang & this causes problems

so if you want to write a scala script:

```
$ hello.scala
```

```
$ chmod a+x hello.scala
```

```
$ scala hello.scala # b/c there's no shebang (inconvenient)
```

you could put `#!SHELL SCRIPT WRAPPER` AROUND

THIS, OR ANY SCRIPT FOR THAT MATTER, TO

WORK AROUND THIS! HERE'S HOW:

```
#!/bin/sh
```

```
exec scala "$@" "$@"
```

```
!#
```

```
def max(.....)
```

```
≡
```

```
!# says  
that scala  
code starts
```

Now

```
$ hello.scala # runs beautifully!
```

```
var i = 0
```

```
while (i < args.length)
```

```
{
```

```
  println(args(i))      ### NORMAL STUFF... BUT
```

```
  i += 1
```

```
}
```

```
args.foreach(arg => println(arg))
```

```
args.foreach(println)
```

VERY NICE, VERY COMPACT

```
for (arg ← args)    # DUDE!  
  println(arg)
```

```
// HELLO WORLD IN JAVA
```

```
class HelloWorld  
{  
  public static void main (String args[])  
  {  
    System.out.println("HelloWorld");  
  }  
}
```

```
// save it , run javac , then java to execute
```

```
// HELLO WORLD IN SCALA
```

```
object HelloWorld  
{  
  def main (args: Array[String]) { # if one line, use '=' sign  
    println("HelloWorld")  
  }  
}
```

```
/* ABBAS ASIDE */
```

```
class HelloWorld  
{  
  static  
  {  
    System.out.println("HelloWorld");  
    System.exit(0); //this is the trick!  
  }  
}
```

```
// WAY TO WRITE JAVA STANDALONE w/OUT MAIN. TRICKY!
```