

Applets

Runs in browser
awt vs. swing (Applet vs Japplet)
init()
start(),stop()
destroy()
security (sandbox).

demo code

Servlets

Server-side (no gui stuff).
- respond to web requests
Efficiency (threads vs. CGI)
Security
Use all the Java APIs.
- including JDBC
Java Server Pages (JSP)

Servlets

HttpServlet
doGet(), doPost() (tied to HTTP)
javax.servlet packages (provided by servlet server
implementation).
sample code
- servlet
- jsp

Network Programming

Communication between processes

Many approaches:

- low level network API: sockets
 need to understand networking, addressing, etc.
- Object oriented approaches: RMI, CORBA
- Service based approaches: URL class.
- *there are others*

Sockets

Sockets API is popular C programming API.

A *socket* is an abstract communication endpoint.

Different kinds of sockets for different kinds of communication:

- Stream (connected byte stream).
- Datagram (individual messages).

To use sockets you need to understand lots of network stuff, including the OSI reference model, network addressing, byte-ordering, ...

Java Sockets

Sockets in Java is much simpler than in C

- hides some of the details.
- API is "fit" to be used with TCP/IP, not as generic.

Different objects used for different kinds of network communication:

- Socket (Stream Client)
- ServerSocket (Stream Server)
- DatagramSocket (Message oriented).

Socket and ServerSocket (TCP based communication)

Establish a *connection* with peer process.

socket provides an IO stream associated with the connection.

- all Java IO methods/Objects can be used.

New exceptions to deal with.

Great it what you really need is to communicate using nothing more than a stream.

Other endpoint could be written in any language!

DatagramSocket and DatagramPacket (UDP)

Message-oriented communication.

- send a message, receive a message
- not dependent on Java IO support...

Lots of issues:

- not reliable.
- order not guaranteed.
- duplication of messages is possible.

Java Sockets and IP Addresses

```
class InetAddress
```

Methods

```
static InetAddress getByName(String);
```

```
static InetAddress getLocalHost();
```

```
byte getAddress();
```

```
String getHostAddress();
```

```
String getHostName();
```

TCP (stream) Client vs. Server

Clients use Socket object

- need to specify the address of the server!
 - IP address and port number
 - can get stream associated with the connection.

Servers use ServerSocket object

- need to set server port number.
- can receive new connections.
 - new connection creates a Socket object.

Sample Talk Client/Server

Talker class:

- *threaded(runnable)*
- *reads from an input stream, writes to an output stream.*

TalkServer class:

- *creates ServerSocket, waits for client to connect.*
- *creates 2 Talker objects and runs them.*

TalkClient class:

- *creates Socket, connects to server.*
- *creates 2 Talker objects and runs them.*

RMI Remote Method Invocation

Basic idea is to make Java objects available over the network.

Server must include code that looks for requests

- at some address (service name).
- call the requested method and return the result.
- serialization!

Client must create remote objects

- after this step there is syntactic difference in the code!

RMI Sample Code

SimpleRMI, AlternativeSimpleRMI

- remote addition, subtraction, ...
- two methods of creating server objects that can service clients.

SortRMI

- remote sort method
- can easily send/receive complex object types.
