

Java Network Programming

Applets

Servlets, JSP

Sockets Programming

RMI

URL

Applets

- GUI program that runs in a web browser.
- Java bytecode comes from a web server (along with a web page).
- Security Issues!
 - sandbox
- Sample applet for those interested...

Servlets

- Server Side Java programming
 - No GUI programming.
 - Dynamic web page creation – responds to web requests.
 - Efficiency issues (threads, persistent objects, etc)
 - Security Issues
 - JSP: Java Server Pages
 - a different way to create servlets – easier than full Java class definitions.

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.
 - really need to understand networking to use effectively.

The Internet Programming

- Every *endpoint* is identified by a triple:
 - Protocol (UDP vs. TCP).
 - Port number (identifies the service)
 - IP address (corresponds to a hostname)
 - Humans typically use hostnames
 - The networking code uses IP addresses
- google.com vs. 64.233.187.99

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 or hostname 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.

URL support

- Uniform Resource Locator: pointer to some resource (typically on the WWW).
- Java has a URL class that can:
 - build and parse URLs
 - create a URLConnection object
 - can interact with the server
 - establish a connection to a URL
 - give you an InputStream so you can read from the server.
 - useful if all you want to do is fetch a web document.

Some URL Methods

```
String getHost();
```

```
String getPath();
```

```
String getQuery();
```

```
InputStream openStream();
```

```
URLConnection openConnection();
```

Fetch and print a web page

```
try {  
    URL u = new URL("http://www.google.com");  
    BufferedReader r = new BufferedReader(  
        new InputStreamReader( u.openStream() ));  
    String s;  
    while ( (s = r.readLine()) != null)  
        System.out.print(s);  
} catch (... I/O and URL Exceptions ..
```