Java Sockets API and Java Threads

November 19, 2004

Outline

- Java Socket API
  - low level API: addresses, sockets, interfaces
  - high level API: URI, URL, connections
- Java Threads
  - synchonization

Java Socket - low level API

- `Socket`
  - TCP client API
  - is used to connect to a remote host
- `ServerSocket`:
  - TCP server API
  - is used to accept connections from client sockets.
- `DatagramSocket`
  - UDP endpoint API
  - is used to send, and receive, `DatagramPackets`.
- `MulticastSocket`
  - subclass of the `DatagramSocket` used when dealing with multicast groups.

`ServerSocket` - Constructors

- `ServerSocket()`
  Creates an unbound server socket.
- `ServerSocket(int port)`
  Creates a server socket, bound to the specified port.
- `ServerSocket(int port, int backlog)`
  Creates a server socket and binds it to the specified local port number, with the specified backlog.
- `ServerSocket(int port, int backlog, InetAddress bindAddr)|`
  Create a server with the specified port, listen backlog, and local IP address to bind to.

`ServerSocket` - Methods

- `Socket accept()`: Listens for a connection to be made to this socket and accepts it.
- `void bind(SocketAddress endpoint)`: Binds the ServerSocket to a specific address (IP address and port number).
- `void bind(SocketAddress endpoint,`
  `int backlog}`
  Binds the ServerSocket to a specific address (IP address and port number).
- `void close()`: Closes this socket.

`ServerSocket` - Methods (cont)

- `InetAddress getInetAddress()`: Returns the local address of this server socket.
- `int getLocalPort()`: Returns the port on which this socket is listening.
- `SocketAddress getLocalSocketAddress()`: Returns the address of the endpoint this socket is bound to, or null if it is not bound yet.
- `boolean isBound()`: Returns the binding state of the ServerSocket.
- `boolean isClosed()`: Returns the closed state of the ServerSocket.

`ServerSocket` - Methods (cont)

- `int getReceiveBufferSize()`: Gets the value of the `SO_RCVBUF` option for this ServerSocket, that is the proposed buffer size that will be used for Sockets accepted from this `ServerSocket`.
- `void setReceiveBufferSize(int size)`: Sets a default proposed value for the `SO_RCVBUF` option for sockets accepted from this `ServerSocket`.
- `boolean getReuseAddress()`: Tests if `SO_REUSEADDR` is enabled.
- `void setReuseAddress(boolean on)`: Enable/disable the `SO_REUSEADDR` socket option.

`ServerSocket` - Methods (cont)

- `int getSoTimeout()`: Retrive setting for `SO_TIMEOUT`.
- `void setSoTimeout(int timeout)`: Enable/disable `SO_TIMEOUT` with the specified timeout, in milliseconds.

`InetAddress` - Methods

- `byte[] getAddress()`: Returns the raw IP address of this InetAddress object.
- `static InetAddress[] getAllByName(String host)|`
  Given the name of a host, returns an array of its IP addresses, based on the configured name service on the system.
- `static InetAddress getByAddress(byte[] addr)|`
- `static InetAddress getByAddress(String host, byte[] addr)`

`InetAddress` - Methods (cont)

- `static InetAddress getByName(String host)`: Determines the IP address of a host, given the host's name.
- `String getCanonicalHostName()`: Gets the fully qualified domain name for this IP address.
- `String getHostAddress()`: Returns the IP address string in textual presentation.
- `String getHostName()`: Gets the host name for this IP address.
- `static InetAddress getLocalHost()`: Returns the local host.

InetAddress - Methods (cont)

```
boolean isAnyLocalAddress()
boolean isLinkLocalAddress()
boolean isLoopbackAddress()
boolean isMCGlobal()
boolean isMCLinkLocal()
boolean isMCNodeLocal()
boolean isMCOrgLocal()
boolean isMCSiteLocal()
boolean isMulticastAddress()
boolean isReachable(int timeout)
boolean isReachable(NetworkInterface netif,
                    int ttl, int timeout)
boolean isSiteLocalAddress()
```

`Socket` - Constructors

- `Socket()`
- `Socket(InetAddress address, int port)`: Creates a stream socket and connects it to the specified port number at the specified IP address.
- `Socket(InetAddress address, int port, InetAddress localAddr, int localPort)` Creates a socket and connects it to the specified remote address on the specified remote port.
- `Socket(Proxy proxy)`
- `Socket(String host, int port)`: Creates a stream socket and connects it to the specified port number on the named host.
- `Socket(String host, int port, InetAddress localAddr, int localPort)`

Socket - Method

```
void bind(SocketAddress bindpoint)

void connect(SocketAddress endpoint)
void connect(SocketAddress endpoint,
             int timeout)

void close()
void shutdownInput()
void shutdownOutput()
```

Socket - Method (cont)

```
InetAddress      getInetAddress()
int              getPort()
InetAddress      getLocalAddress()
int              getLocalPort()
SocketAddress    getLocalSocketAddress()
SocketAddress    getRemoteSocketAddress()

InputStream      getInputStream()
OutputStream     getOutputStream()
void             sendUrgentData(int data)
```

Socket - Method (cont)

```
boolean getKeepAlive()
void    setKeepAlive(boolean on)
boolean getOOBInline()
void    setOOBInline(boolean on)

int     getReceiveBufferSize()
void    setReceiveBufferSize(int size)
int     getSendBufferSize()
void    setSendBufferSize(int size)
```

Socket - Method (cont)

```
boolean getReuseAddress()
void    setReuseAddress(boolean on)
int     getSoLinger()
void    setSoLinger(boolean on, int linger)
int     getSoTimeout()
void    setSoTimeout(int timeout)
boolean getTcpNoDelay()
void    setTcpNoDelay(boolean on)
int     getTrafficClass()
void    setTrafficClass(int tc)
```

Socket - Method (cont)

```
boolean isBound()
boolean isClosed()
boolean isConnected()
boolean isInputShutdown()
boolean isOutputShutdown()
```

`DatagramSocket` - Constructors

- `DatagramSocket()`: Constructs a datagram socket and binds it to any available port on the local host machine.
- `DatagramSocket(int port)`: Constructs a datagram socket and binds it to the specified port on the local host machine.
- `DatagramSocket(int port, InetAddress laddr)`
  Creates a datagram socket, bound to the specified local address.
- `DatagramSocket(SocketAddress bindaddr)`: Creates a datagram socket, bound to the specified local socket address.

DatagramSocket - Methods

```
void bind(SocketAddress addr)
void close()
void connect(InetAddress address, int port)
void connect(SocketAddress addr)
void disconnect()
```

DatagramSocket - Methods

```
void  receive(DatagramPacket p)
void  send(DatagramPacket p)

boolean getBroadcast()
void    setBroadcast(boolean on)

boolean isBound()
boolean isClosed()
boolean isConnected()
```

`DatagramPacket` - Constructors

- `DatagramPacket(byte[] buf, int length)`
  Constructs a `DatagramPacket` for receiving packets of length `length`.
- `DatagramPacket(byte[] buf, int length, InetAddress address, int port)`
  Constructs a datagram packet for sending packets of length length to the specified port number on the specified host.
- `DatagramPacket(byte[] buf, int offset, int length)`
  Constructs a `DatagramPacket` for receiving packets of length length, specifying an offset into the buffer.

`DatagramPacket` - Constructors(cont)

- `DatagramPacket(byte[] buf, int offset, int len, InetAddress addr, int port)`

  Constructs a datagram packet for sending packets of length length with offset `offset` to the specified port number on the specified host.

- `DatagramPacket(byte[] buf, int offset, int len, SocketAddress addr)`

- `DatagramPacket(byte[] buf, int len, SocketAddress addr)`

Other components

- `URI` is the class representing a Universal Resource Identifier, as specified in RFC 2396. As the name indicates, this is just an Identifier and doesn't provide directly the means to access the resource.
- `URL` is the class representing a Universal Resource Locator, which is both an older concept for URIs and a mean to access the resources.
- `URLConnection` is created from a `URL` and is the communication link used to access the resource pointed by the `URL`. This abstract class will delegate most of the work to the underlying protocol handlers like http or ftp.
- `HttpURLConnection` is a subclass of `URLConnection` and provides some additional functionalities specific to the HTTP protocol.

“There are two ways of constructing a software design; one way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.” – C. A. R. Hoare

Syncronization

- Mutual Exclusion Locks: mutex
- Semaphores (E. W. Dijkstra)
- Monitors (C. A. R. Hoare)

Java suport for threading

- class `Thread`
- interface `Runnable`
- `synchronized`
- `wait()`
- `notify`, `notifyAll()`

... to be continued.