



JavaOneSM
Sun's 2003 Worldwide Java Developer Conference™

Java™ Technology- Based Networking: Advanced Concepts and Troubleshooting

Jean-Christophe Collet

Michael McMahon

Yingxian Wang

Java Security & Networking
Sun Microsystems

Primary Purpose

Learn how to use efficiently the networking API and know what's coming in JDK™ 1.5.0

Speaker's Qualifications

- Jean-Christophe Collet has been with the Security and Networking group for Java™ technology for more than 3 years
- Michael McMahon for more than 2 years
- Yigxian Wang for 4 years

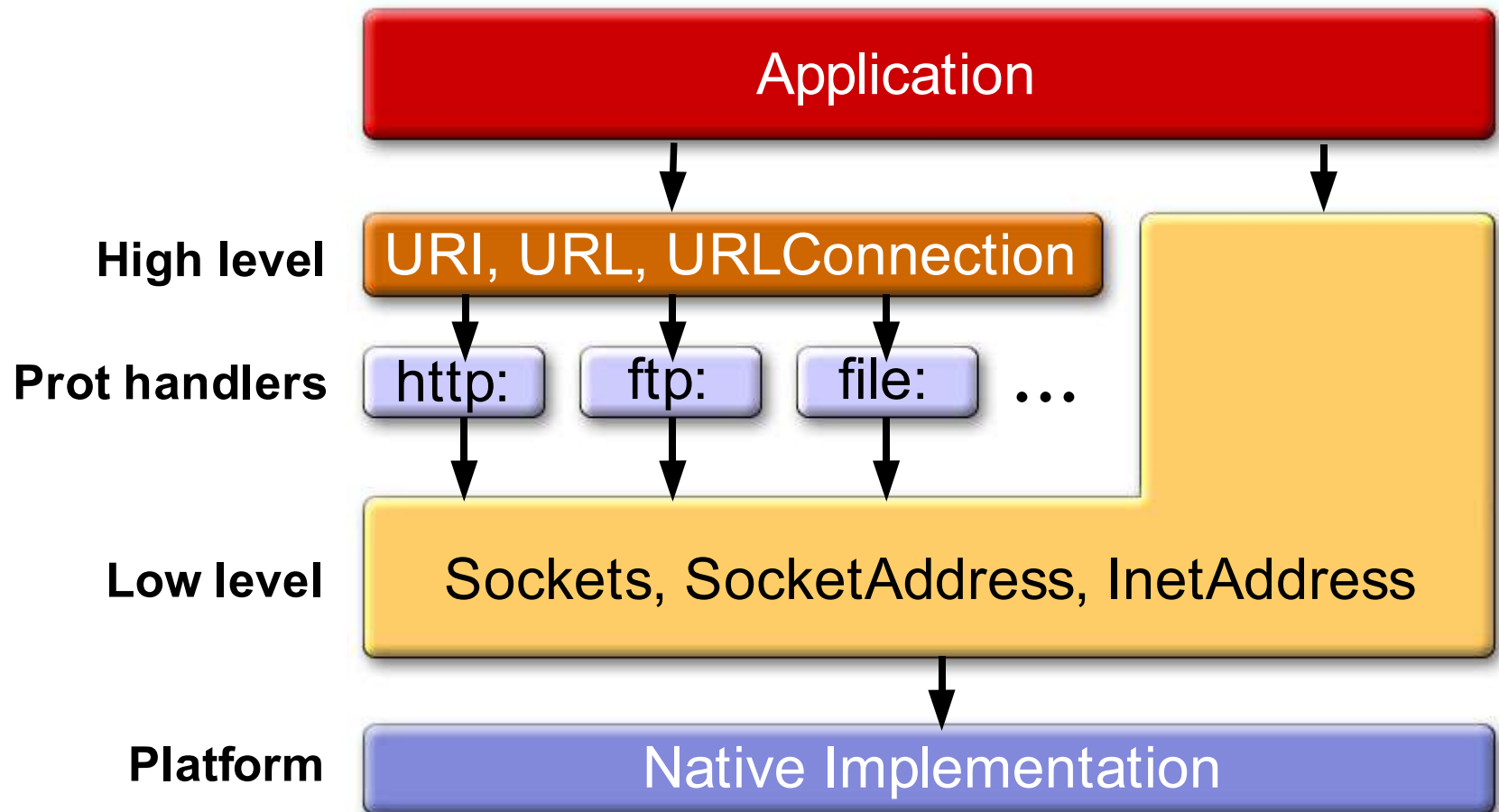
Agenda

- Introduction
- Java.net API overview
- URIs, URLs and URLConnections
- HTTP and FTP protocol handlers
- Name resolution gotchas
- J2SE™ 1.5.0 specification new features
- Conclusion
- Q/A

Introduction

- Security and Networking group for Java™ technology
- Networking team
 - Jean-Christophe Collet, Technical Lead
 - Michael McMahon, Engineer
 - Yingxian Wang, Engineer
- Responsible for:
 - java.net (public API)
 - sun.net (protocol handlers)

java.net API Overview



What About java.nio?

- Low level non blocking networking API
 - Channels
 - Selectors
- Intended for scalable server applications
- More complex
- Sessions 2180–2182, Wed. 4:00pm & 5:15pm, G102
- BOF 2185, Thurs. 7:30 pm, Argent Hotel, Met III conference room

URI, URL and URLConnection

- URI
 - Universal Resource Identifier
 - For parsing
- URL
 - Universal Resource Locator
 - For fetching
- URLConnection
 - Finer control of the transfer
- HttpURLConnection
 - Subclass specific to HTTP

Example: Simple http GET

```
URI uri = new URI("http://www.sun.com/");
System.out.println("Scheme: " + uri.getScheme());

// Convert into a URL for transfer
URL url = uri.toURL();

// Open the connection
URLConnection conn = url.openConnection();
BufferedReader in = new BufferedReader(
    new InputStreamReader(conn.getInputStream()));

// What size is the document?
int len = conn.getContentLength();
System.out.println("Content Length: " + len);

// Print the document
String line;
while ((line = in.readLine()) != null)
    System.out.println(line);
in.close();
```

Example: Cookie Handling

```
URL url = new URL("http://host.mydom/");
URLConnection conn = url.openConnection();

// Add cookies from cache to the request

conn.addRequestProperty("Cookie", cookie1);
conn.addRequestProperty("Cookie", cookie2);

// Send the GET request

InputStream in = conn.getInputStream();

// Get cookies from response headers

List cookies = (List)
    conn.getHeaderFields().get("Cookie");
```

Example: HEAD Request

```
URL url = new URL("http://www.sun.com/");
URLConnection http = (URLConnection)
    url.openConnection();

// Make it a HEAD Request
http.setRequestMethod("HEAD");

// Get the response code
int ret = http.getResponseCode();
if (ret == http.HTTP_OK) {
    int size = http.getHeaderFieldInt("Content-
length", 0);
    System.out.println("length = " + size);
    System.out.println("last modified:" +
        http.getHeaderField("Last-Modified"));
}
```

Example: Error Handling

```
try {
    URL url =
        new URL("http://java.sun.com/badFile");
    HttpURLConnection http = (HttpURLConnection)
        url.openConnection();
    InputStream is = http.getInputStream();
    is.read(buf);
    // Do stuff
} catch (IOException e) {
    int respCode = http.getResponseCode();
    InputStream es = http.getErrorStream();
    if (respCode == http.HTTP_NOT_FOUND &&
        es != null) {
        es.read(buf);
        // Manage error
    }
}
```

Example: POST Request

```
URL url = new URL("http://host.mydom/printenv");
URLConnection http = (URLConnection)
    url.openConnection();

// We want to send a POST request
http.setDoOutput(true);
PrintWriter out = new
    PrintWriter(http.getOutputStream());
out.println("Blablabla");
out.close();

BufferedReader in = new BufferedReader(new
    InputStreamReader(http.getInputStream()));
String line;
while ((line = in.readLine()) != null)
    System.out.println(line);
in.close();
```

HTTP Handler: Summary

- **URLConnection**
 - Simple
 - Enough for most general uses (GET & POST)
 - Still some flexibility
- **HttpURLConnection**
 - A bit more complex
 - More flexibility (response code, RequestMethod)
 - Needs an explicit cast
- **Http protocol handler**
 - Keep-alive, Redirection & Authentication

HTTP Handler: Keep-alive

- Tries to use keep-alive where possible
- System property (boolean):
`http.keepAlive`
- Response from server can determine details of connection re-use
- Disconnect() method prevents reuse
- Application **MUST** call close() on stream

HTTP Handler: Redirection

- Default policy set with
`setFollowRedirects()`
- Per instance policy set with
`setInstanceFollowRedirects()`
- Redirection between http and https never automatic
- POST can be converted to GET
`http.strictPostRedirect`

HTTP Authentication

- Client to origin server, or proxy (or both)
- Supported schemes:
 - Basic (older, not secure)
 - Digest (most secure)
 - NTLM (since 1.4.2, on Windows only)
- Supported protocols: http and https
- `java.net.Authenticator` abstract class
 - Needs to be subclassed

Authenticator Example

```
public class myAuth extends Authenticator {
    PasswordAuthentication pw;

    public PasswordAuthentication
        getPasswordAuthentication () {
        // interact with user to get user, pass
        pw = new PasswordAuthentication(user,
            pass.toCharArray());
        return pw;
    }
}
```

// In the application

```
Authenticator auth = new MyAuth();
Authenticator.setDefault(auth);
```

Authenticator: Hints and Tips

- Use Authenticators concrete methods to interact with user:

```
getRequestingHost()  
    ↖ "www.foo.com"  
getRequestingProtocol()  
    ↖ "http", "https" or "SOCKS"  
getRequestingPrompt()  
    ↖ "foo.com order entry"  
getRequestingScheme()  
    ↖ "basic", "ntlm", "digest"
```

- Specials for NTLM
 - separate entry field for NT domain name
 - encode domain in user field as: "domain\user"

FTP : The Forgotten Protocol

- There is a protocol handler for FTP!
- URL format defined in RFC 1738
- Functionalities:
 - GET, PUT, DIR
 - Passive Mode, IPv6 extensions
 - Username / Password authentication
- Examples:

```
ftp://user:passwd@mysite.com/pub/files.txt?type=a  
ftp://othersite.com/%2Fetc?type=d
```

FTP Example

```
// List a directory
```

```
URL url = new URL("ftp://host.domain/pub/");
URLConnection conn = url.openConnection();
BufferedReader in = new BufferedReader(
    new InputStreamReader(conn.getInputStream()));
String line;
while ((line = in.readLine()) != null)
    System.out.println(line);
in.close();
```

```
// Put a file with authentication
```

```
url =
new URL("ftp://user:passwd@host.domain/%2Fetc/motd");
conn = url.openConnection();
PrintWriter out = new PrintWriter(conn.getOutputStream
());
out.println("Message of the day");
out.close();
```

Name Resolution

- InetAddress class
- Forward (hostname to address):
 - InetAddress.getByName:

```
InetAddress a =  
    InetAddress.getByName ("www.sun.com") ;
```

- Reverse (address to hostname)
 - InetAddress.getCanonicalHostName:

```
InetAddress a =  
    InetAddress.getByName ("192.168.1.1") ;  
String host = a.getCanonicalHostName () ;
```

- Default: System resolver, but JNDI/DNS possible

Name Resolution and Caching

- InetAddress has a cache
 - For security (e.g. DNS spoofing)
 - Can help with performance
- Default behavior:
 - Positive (successful): no expiration
 - Negative (unresolved): 10 sec
- System properties (values in seconds):
 - `networkaddress.cache.ttl` (default -1)
 - `networkaddress.cache.negative.ttl` (default 10)

J2SE™ 1.5 Specification Goals

HttpURLConnection improvements

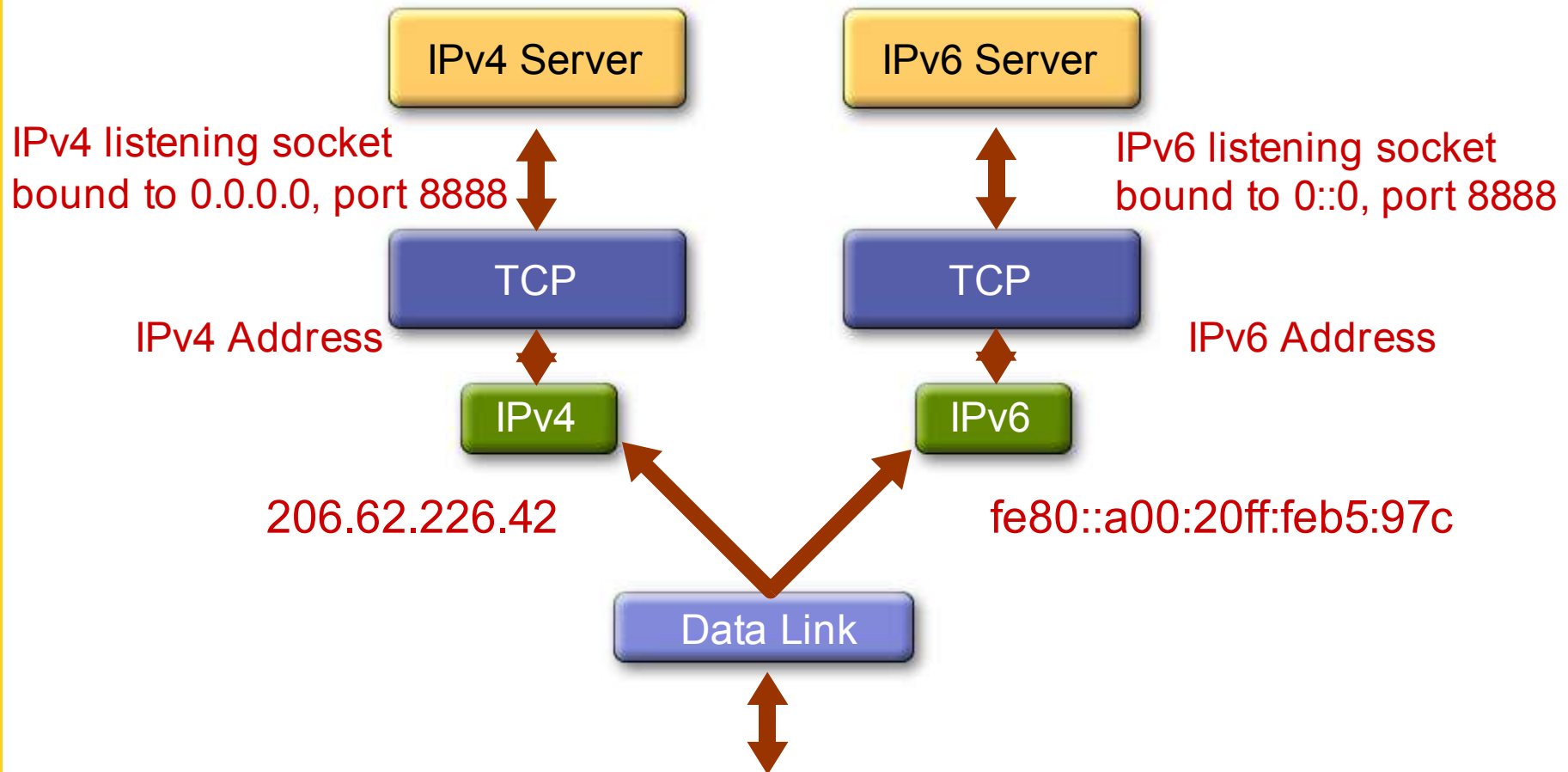
- Faster implementation (handler)
- Read and connect timeouts
- Cookie management
- Proxy management
- Pipelined requests
- Support for streaming

J2SE 1.5 Specification Goals

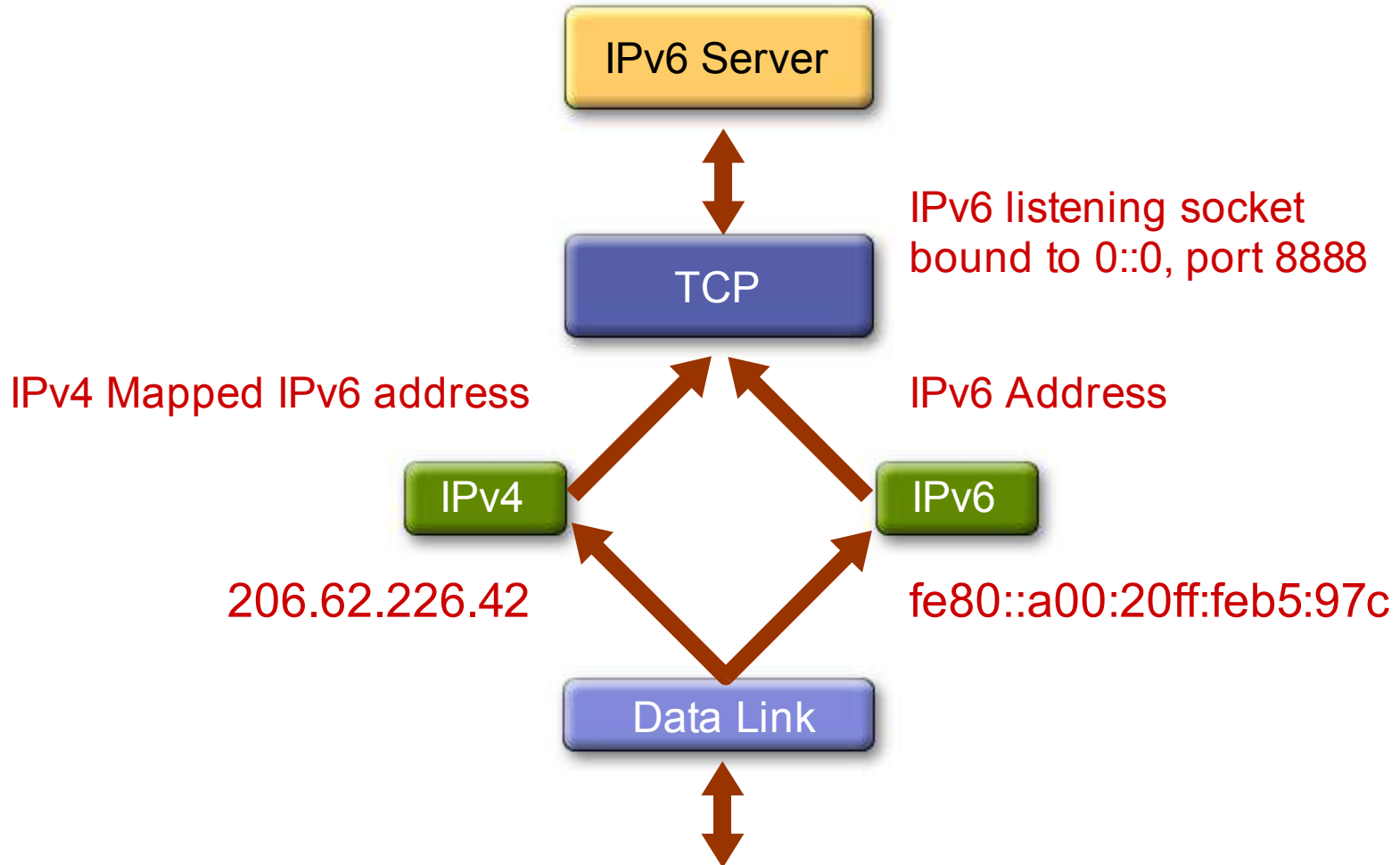
IPv6 improvements

- Windows XP SP1 and Windows 2003 support
- Hides the 2 separate stacks
- Transparent and automatic
- Every Java™ application is already IPv6 ready
- Some QoS options

Separate Stacks System



Dual Stack System



Separate Stacks: Simplified Code Example (in C)

```
SOCKET ServSock[FD_SETSIZE];
ADDRINFO AI0, AI1;
ServSock[0] = socket(AF_INET6, SOCK_STREAM, PF_INET6);
ServSock[1] = socket(AF_INET, SOCK_STREAM, PF_INET);
...
bind(ServSock[0], AI0->ai_addr, AI0->ai_addrlen);
bind(ServSock[1], AI1->ai_addr, AI1->ai_addrlen);
...
select(2, &SockSet, 0, 0, 0);
if (FD_ISSET(ServSocket[0], &SockSet)) {
    // IPv6 connection
    csock = accept(ServSocket[0], (LPSOCKADDR) &From,
                  FromLen);
    ...
}
if (FD_ISSET(ServSocket[1], &SockSet)) {
    // IPv4 connection
    csock = accept(ServSocket[1], (LPSOCKADDR) &From,
                  FromLen);
    ...
}
```

Dual Stack Code Example (Java)

```
ServerSocket server =  
    new ServerSocket(port);  
Socket s;  
while (true) {  
    s = server.accept();  
    doClientStuff(s);  
}
```

IPv6 Summary

- Transparent IPv6 support in java.net:
 - Since 1.4.0 for Solaris 8+ and Linux
 - In 1.5.0 for Windows XP SP1 and Windows 2003
- No code change
- No Recompile
- We've done all the hard work for you. Use it!

Summary

- 2 levels of abstraction in java.net
 - High level: URLs
 - Low level: Sockets
- HttpURLConnection can be very useful
- 1.5 will improve http support
- With Java™ applications you get IPv6 connectivity for free
- BOF 2240: Thursday 10:30pm, Argent Hotel, Concordia room
- java-net@sun.com

Primary Purpose

Learn how to use efficiently the networking API and know what's coming in JDK™ 1.5.0

Q&A

JAVA™



JavaOneSM

Sun's 2003 Worldwide Java Developer Conference*

JAVATM

java.sun.com/javaone/sf