



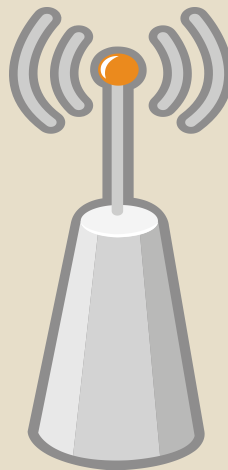
Chapter 11

Java Network Programming



javaseu@163.com

Xiang zhang





Content

2

- Identification of Machine
- Client and Server
- Port
- Java Socket Programming Primer (C/S Programming on TCP)
- Java Web Enlightenment



Identification of Machine

3

- IP address - Internet Protocol
 - Domain name or host name: cose.seu.edu.cn
 - Four fragments: 58.192.114.215
- IPv4
 - A figure in 32-bits
 - Almost 4,000,000,000 IPs
- IPv6
 - A figure in 128-bits
 - Guess how many IPs?



Identification of Machine

4

- IP

- 58.192.112.11
- Identical in WAN or in LAN

- Hostname

- Spark-pc
- Identical in LAN

- Domain Name

- www.seu.edu.cn
- Translated into IP by using DNS



```
public void getIP(){
    try{
        //得到InetAddress
        InetAddress iAddress = InetAddress.getLocalHost();
        //获得本机IP
        String localIP = iAddress.getHostAddress().toString();
        //获得本机名称
        String hostName=iAddress.getHostName().toString();
        System.out.println("您的IP为：" + localIP);
        System.out.println("您的主机名为：" + hostName);
    }catch(UnknownHostException e){
        e.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }
}
```



Getting IP of Multiple Network Adaptor

/ 通过本机的主机名获取所有IP */*

```
public ArrayList<String> getAllIP(){
    ArrayList<String> allIP = new ArrayList<String>();
    try{
        String hostName = InetAddress.getLocalHost().getHostName();
        if(hostName.length()>0){
            InetAddress[] addresses = InetAddress.getAllByName(hostName);
            for(int i=0; i<addresses.length; i++){
                allIP.add(addresses[i].getHostAddress().toString());
            }
        }
        return allIP;
    }catch(Exception e){
        e.printStackTrace();
        return allIP;
    }
}
```



InetAddress Class

7

- Constructor localhost InetAddress

```
InetAddress addr = InetAddress.getByName(null);  
InetAddress addr = InetAddress.getByName("127.0.0.1");  
InetAddress addr = InetAddress.getByName("localhost");  
InetAddress addr = InetAddress.getLocalHost();
```

```
byte[] ip = {127,0,0,1};  
InetAddress addr = InetAddress.getByAddress(ip);
```

- Construct InetAddress of other machine

```
InetAddress addr = InetAddress.getByName("cose.seu.edu.cn");
```

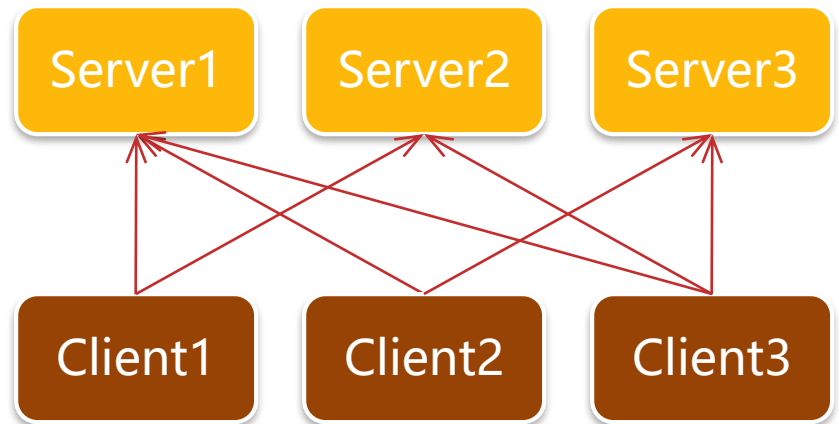
```
byte[] ip = {(byte)58,(byte)192,(byte)114,(byte)215};  
InetAddress addr = InetAddress.getByAddress(ip);
```

```
private static final int TIMEOUT = 5000;
public void ping(InetAddress addr){
    try{
        String hostName = addr.getHostName();
        while(true){
            if(addr.isReachable(TIMEOUT)){
                System.out.println("Reply from "
                    + hostName + " within " + TIMEOUT + "ms.");
            }
            Thread.sleep(1000);
        }
    }catch(Exception e){e.printStackTrace();}
}
```




Client and Server

- Server
 - Response passively, intercepting requests
- Client
 - Request actively





Port

10

- IP identifies machines, but cannot identify apps
- Considering our server:
 - Web server <http://cose.seu.edu.cn>
 - FTP server <ftp://cose.seu.edu.cn>
 - Mail server <smtp://mail.seu.edu.cn>
- IP – House number; Port – Room number
- Client communicates with a port on server
- Port 1-1024 is occupied



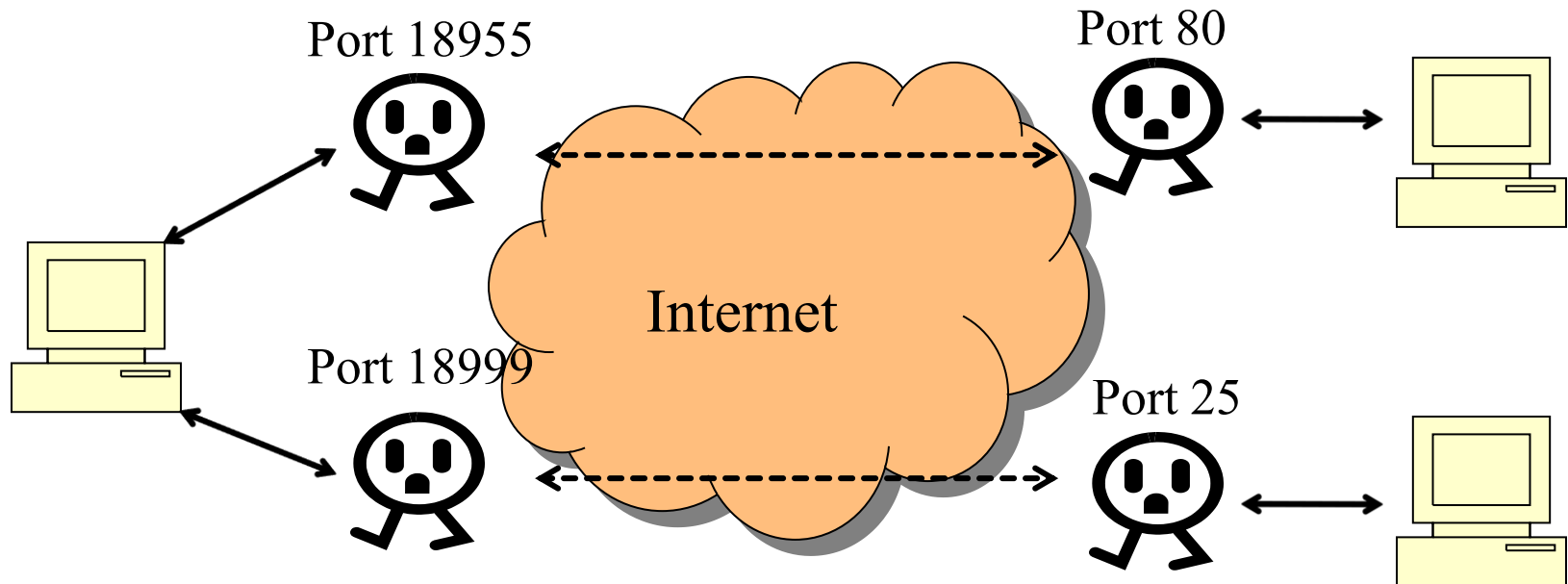
Occupied Port

11

Port	Service
21	FTP
23	TELNET
25	SMTP
53	DNS
80	HTTP
110	POP3
1080	SOCK



- A **virtual terminal** between two machines for a connection
- Data flows from one Socket to the other





Socket



13

- Client create a Socket to connection to server

//通过host name来构建客户端Socket

```
Socket client = new Socket("cose.seu.edu.cn", 8080);
```

//或者通过InetAddress来构建客户端Socket

```
InetAddress address = InetAddress.getByName("cose.seu.edu.cn");
```

```
Socket client = new Socket(address, 8080);
```

- Server create a ServerSocket to intercept request

//创建服务端的ServerSocket监听客户端请求

```
ServerSocket server = new ServerSocket(8080);
```

//当没有客户端请求时，服务器端阻塞，

//当客户端请求到来时，accept()方法将创建一个服务器端Socket

```
Socket serverSocket = server.accept();
```



Socket



14

- Socket – read and write

- Client write data to Socket by OutputStream
- Server read data from Socket by InputStream

```
Socket client = new Socket("cose.seu.edu.cn", 8080);  
InputStream is = socket.getInputStream();  
OutputStream os = socket.getOutputStream();
```

- Remember to close input and output stream, and the Socket itself, after communication.

```
ServerSocket server = new ServerSocket(8088);
System.out.println("服务器已经启动.");
Socket socket = server.accept();
try{
    BufferedReader in = new BufferedReader(
        new InputStreamReader(socket.getInputStream()));
    PrintWriter out = new PrintWriter(new BufferedWriter(
        new OutputStreamWriter(socket.getOutputStream())),true);
    while(true){
        String str = in.readLine();
        if (str!=null && str.equals("你好")) out.println("你好，我是服务器");
        else out.println("听不懂");
    }
}catch(Exception e){
    e.printStackTrace();
}finally{
    socket.close();server.close();
}
```

```
Socket socket = new Socket("localhost", 8088);
BufferedReader in = new BufferedReader(
    new InputStreamReader(socket.getInputStream()));
PrintWriter out = new PrintWriter(
    new BufferedWriter(new OutputStreamWriter(
        socket.getOutputStream())));
out.println("你好");
Thread.sleep(1000);
out.println("今天星期几? ");
socket.close();
```

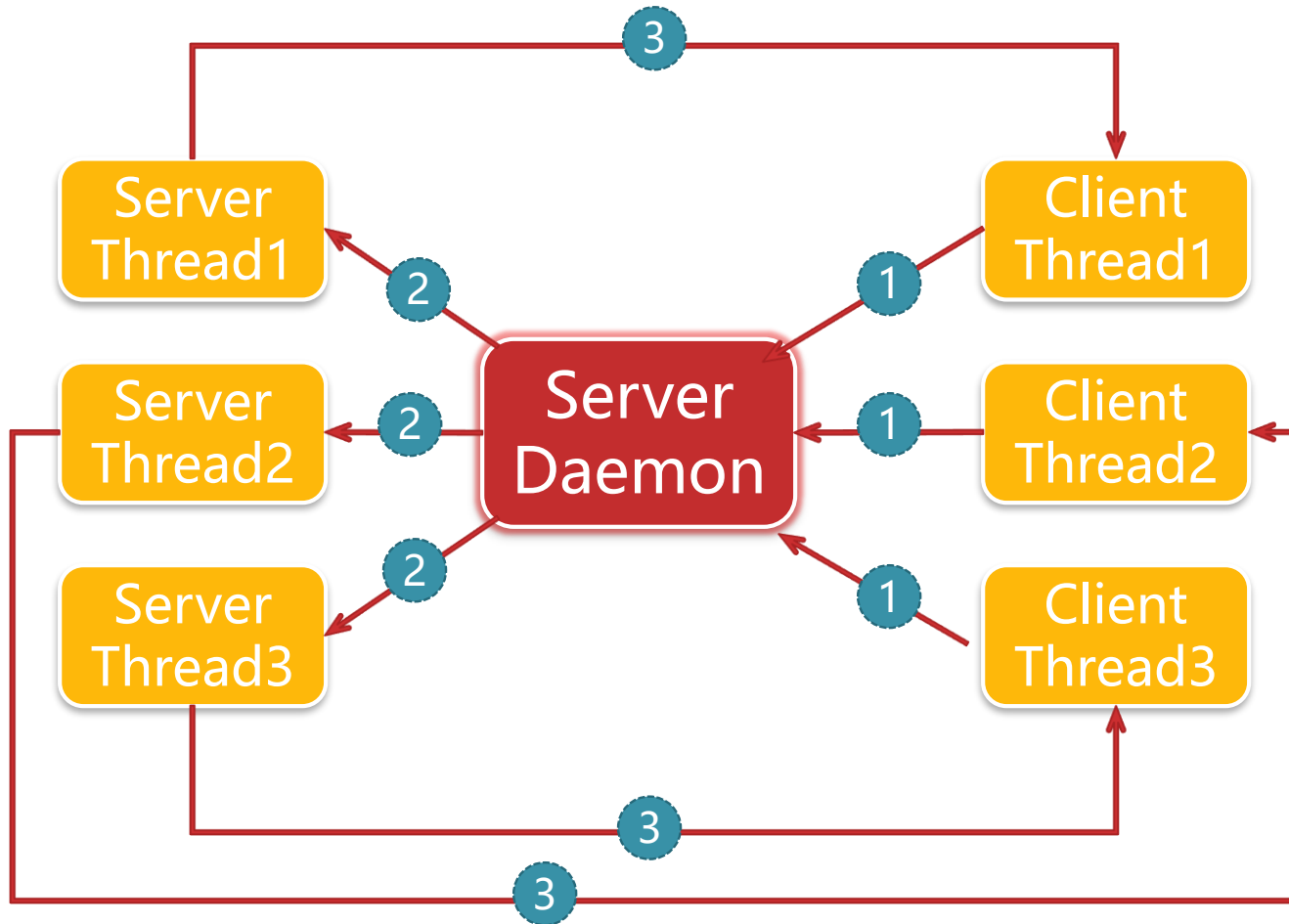
**如果关闭client, 再重启client (不关闭服务器端) ,
客户端会收到什么样的响应?**



TCP and UDP

17

- TCP
 - Based on connection
 - Using data stream to communicate
 - Secured, low possibility to lose data
- UDP
 - No connection
 - Using data packet to communicate
 - Not secured, possible to lose data
 - Self study: using `java.net.DatagramSocket`



- Try to program
 - Create a Daemon on server
 - When requested, Daemon create a thread to response.
 - At most 10 concurrent responding threads.
 - Client tells server its name, then server make a greet. After a Bye said by client, corresponding thread on server exits.
 - A client starts 12 threads to request server, each thread staying at least 3s.

```
public class ServerDaemon {
    public static final int PORT = 8080;
    // 用于控制最大可用线程数
    private static final int MAX_THREADS = 10;
    //用于记录当前已创建线程数
    public static int CURRENT_THREADS = 0;
    ServerSocket server;

    public ServerDaemon(){
        System.out.println("Server started.");
        try{server = new ServerSocket(PORT);
            while(true){
                if(CURRENT_THREADS<MAX_THREADS){
                    //创建新的线程相应客户端请求
                    ServerThread thread = new ServerThread(server.accept());
                    thread.start();}}}
        catch(Exception e){e.printStackTrace(); }
        finally{
            try{server.close();}catch(Exception e){e.printStackTrace();}
        }
    }
}
```

```
public class ServerThread extends Thread{
    Socket socket;
    BufferedReader in;
    PrintWriter out;

    public ServerThread(Socket socket) throws IOException{
        ServerDaemon.CURRENT_THREADS++; //记数
        this.socket = socket;
        //创建BufferedReader用于输入
        in = new BufferedReader(
            new InputStreamReader(this.socket.getInputStream()));
        //创建PrintWriter用于输出
        out = new PrintWriter(new BufferedWriter(
            new OutputStreamWriter(this.socket.getOutputStream())), true);
    }
}
```

```
public void run(){
    try{
        while(true){
            String str = in.readLine();
            if(str!=null){
                if(str.equals("bye")){
                    break;
                }else{
                    String greeting = "Hello " + str + ", I am Server.";
                    out.println(greeting);
                }
            }
        }
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        try{
            socket.close();
            ServerDaemon.CURRENT_THREADS--; //减少记数
        }catch(Exception e){e.printStackTrace();}}
}
```

```
public class ClientThread extends Thread{
    Socket client;
    BufferedReader in;
    PrintWriter out;

    public ClientThread(){
        try{
            InetAddress address = InetAddress.getLocalHost();
            client = new Socket(address, ServerDaemon.PORT);
            in = new BufferedReader(
                new InputStreamReader(client.getInputStream()));
            out = new PrintWriter(new BufferedWriter(
                new OutputStreamWriter(client.getOutputStream())), true);
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

```
public void run(){
    String threadName = Thread.currentThread().getName();
    out.println(threadName);
    try{
        System.out.println(in.readLine());
        Thread.sleep(3000);
        out.println("bye");
    }catch(Exception e){e.printStackTrace(); }
    finally{
        try{client.close();}catch(Exception e){e.printStackTrace();}
    }
}

public static void main(String[] args){
    for(int i=0; i<12; i++){
        ClientThread client = new ClientThread();
        client.start();
    }
}
```



```
ClientThread [Java Application] C:\Program Files\Java
```

```
Sun Dec 29 12:10:19 CST 2019
```

```
Hello Thread-0, I am Server.
```

```
Hello Thread-1, I am Server.
```

```
Hello Thread-2, I am Server.
```

```
Hello Thread-3, I am Server.
```

```
Hello Thread-4, I am Server.
```

```
Hello Thread-5, I am Server.
```

```
Hello Thread-6, I am Server.
```

```
Hello Thread-7, I am Server.
```

```
Hello Thread-8, I am Server.
```

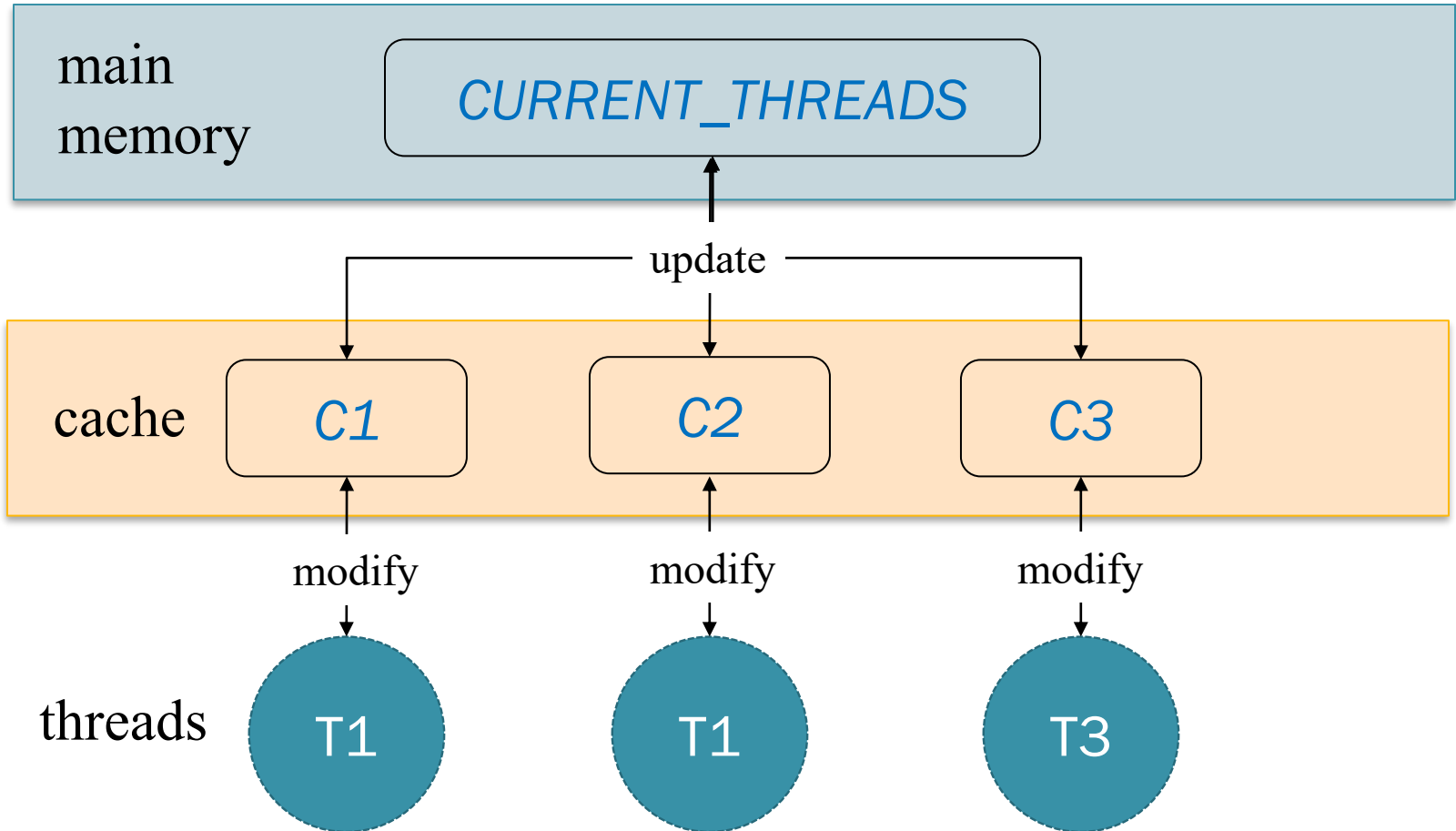
```
Hello Thread-9, I am Server.
```



On the client side, why the program cannot continue with **Thread-10** and **Thread-11**?

In ServerDaemon, the while(true) loop contains a single if() statement, but no other statements. And *CURRENT_THREADS* is a shared static variable.

```
public ServerDaemon(){
    System.out.println("Server started.");
    try{
        server = new ServerSocket(PORT);
        while(true){
            if(CURRENT_THREADS < MAX_THREADS){
                System.out.println("Server daemon with " + CURRENT_THREADS + "threads");
                ServerThread thread = new ServerThread(server.accept());
                thread.start();
            }
        }
    }catch(Exception e){
```



The while(true) trap

```
while(true) {  
    // single statement is a trap!!!  
}
```



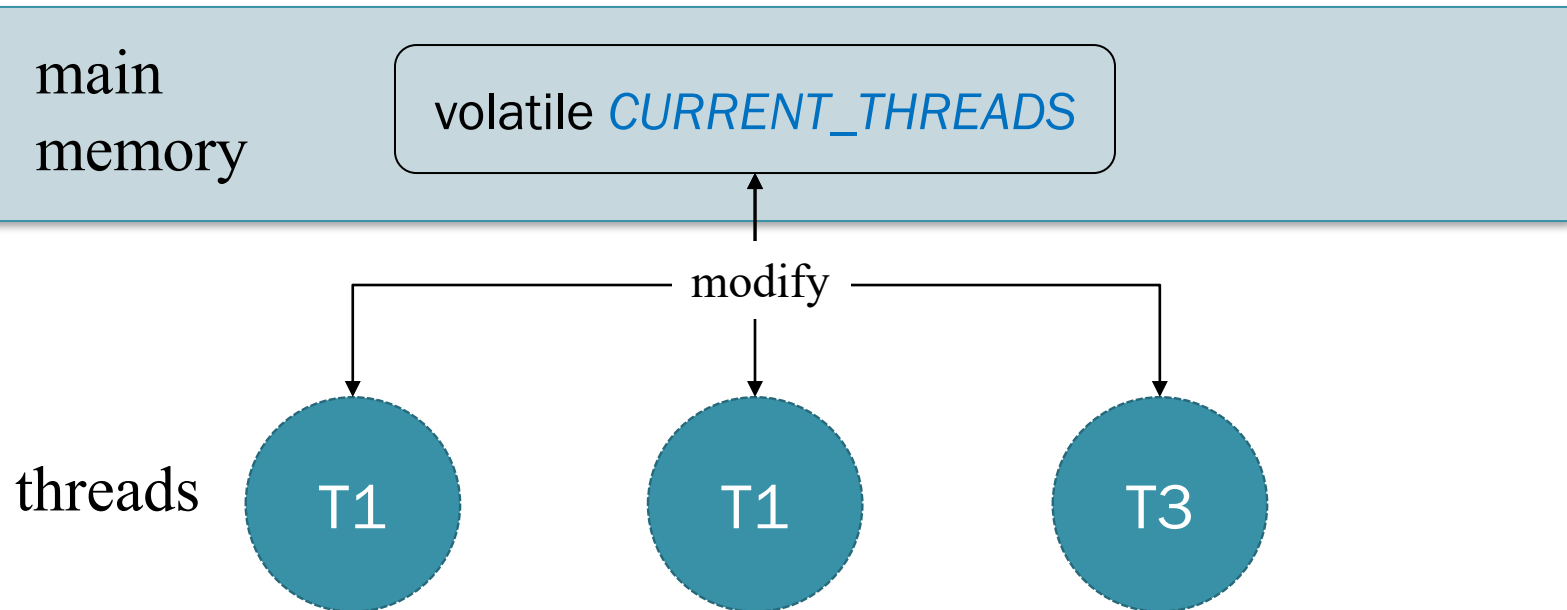
Let the while(true) loop take a break, this is called **Poll**

```
while(true){  
    if(CURRENT_THREADS<MAX_THREADS){  
        System.out.println("Server daemon with " + CURRENT_THREADS + "threads");  
        ServerThread thread = new ServerThread(server.accept());  
        thread.start();  
    }  
    Thread.sleep(1);  
}
```

Or anything like System.out.println("...");

```
public volatile static int CURRENT_THREADS = 0;
```

which means this shared variable should be always up-to-date, and it is identical for each thread that access it.





Java Concurrent

31

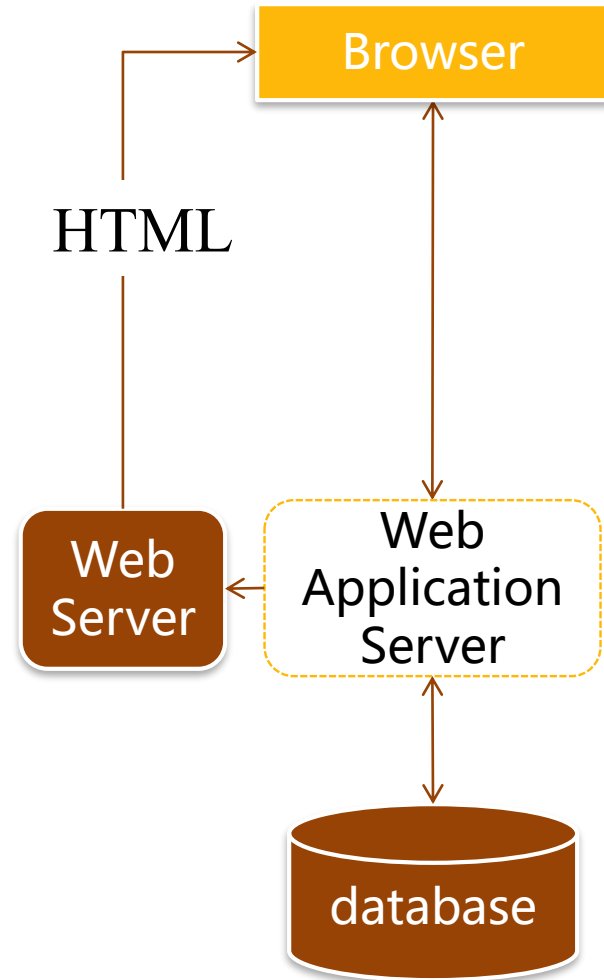
<https://www.cnblogs.com/dolphin0520/p/3920373.html>



Java Web Enlightenment

32

- Web server is on Port 80
 - Apache
 - nginx
 - IIS
- Web App server
 - Tomcat
 - Jboss
 - WebSphere





Java Web Enlightenment

33

- Web front
 - JSP / Servlet
 - RIA(Rich Internet Application)
 - ✦ JavaScript / AJAX
 - ✦ JavaFX
 - ✦ Flex
 - ✦ Silverlight
- Web backend (middleware)
 - Heavy-weighted J2EE – EJB
 - Light-weighted J2EE – Spring / Struts / Hibernate

```
try{
    URL coseURL = new URL("http://cose.seu.edu.cn");
    URLConnection connection = coseURL.openConnection();
    BufferedReader in = new BufferedReader(
        new InputStreamReader(connection.getInputStream()));
    String html = in.readLine();
    while(html!=null){
        System.out.println(html);
        html = in.readLine();
    }
}catch(Exception e){
    e.printStackTrace();
}
```



So easy
!



Self-study

35

- HTML syntax
- Parsing HTML
 - <http://www.open-open.com/30.htm>
- Installing Apache Tomcat
 - Write your personal page, and test it



Thanks

36

Each ending leads to a beginning.

x.zhang@seu.edu.cn