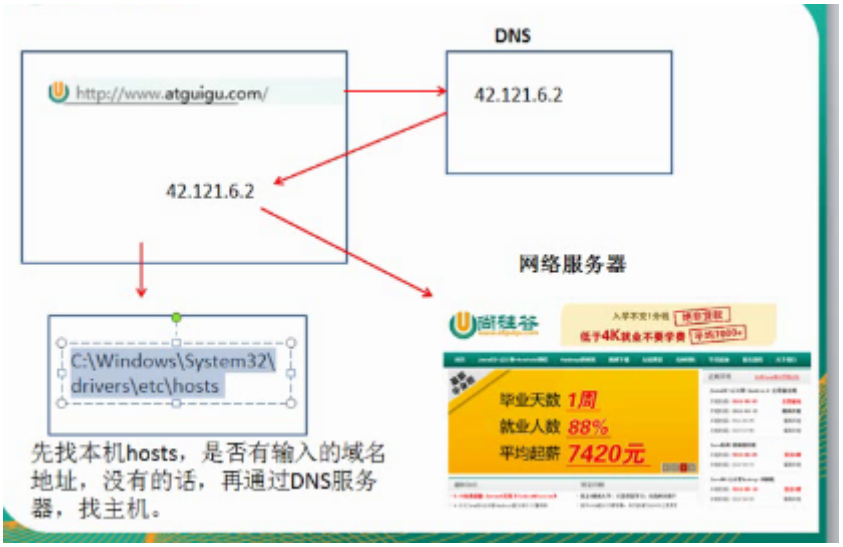



```
* InetAddress:位于java.net包下
* 1.InetAddress用来代表IP地址。一个InetAdress的对象就代表着一个IP地址
* 2.如何创建InetAddress的对象：getByName(String host)
* 3.getHostName()：获取IP地址对应的域名
*   getHostAddress():获取IP地址
```

dns域名解析服务器；

本机IP：

```
Inet4Address inet4Address2  = (Inet4Address) Inet4Address.getLocalHost();
```

其他机器IP：

```
Inet4Address  inet4Address  =  (Inet4Address)
Inet4Address.getByName("www.baidu.com");
```

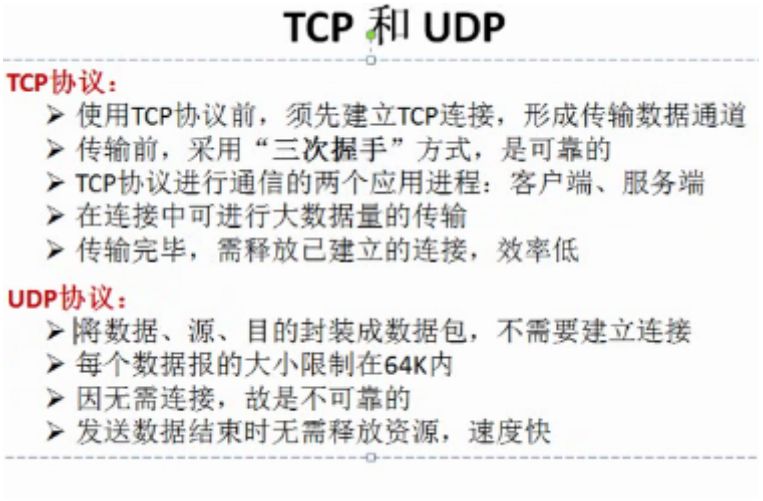
通讯要素2：网络通信协议

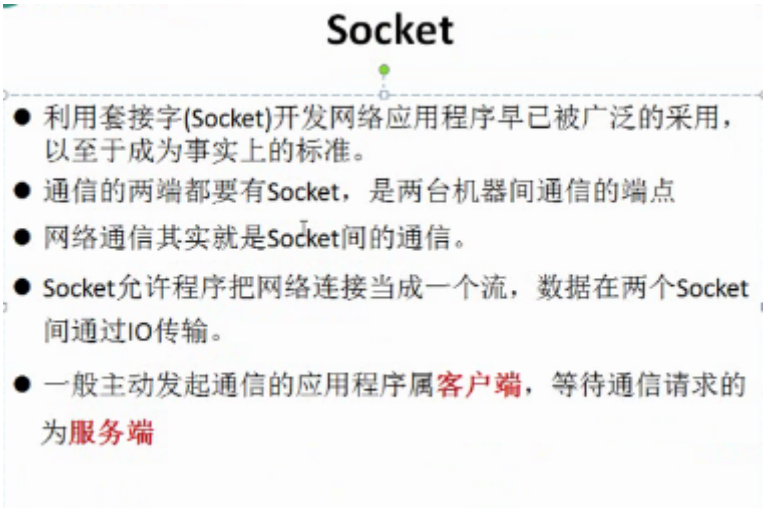
- **网络通信协议**

  计算机网络中实现通信必须有一些约定，即通信协议，对速率、传输代码、代码结构、传输控制步骤、出错控制等制定标准。

- **通信协议分层的思想**

  由于结点之间联系很复杂，在制定协议时，把复杂成份分解成一些简单的成份，再将它们复合起来。最常用的复合方式是层次方式，即同层间可以通信、上一层可以调用下一层，而与再下一层不发生关系。各层互不影响，利于系统的开发和扩展。





网络编程实际上就是socket编程

```
package thread;

import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;

import org.junit.Test;

public class servers {
@Test
public void client() {
        Socket socket = null;
        OutputStream oStream = null;
        try {
                //1，创建一个socket的对象，通过构造器指明服务器的IP地址，以及其接受程序的端口号
                 socket = new Socket(InetAddress.getByName("127.0.0.1"),9090);
                 //2，getOutputstream（）发送数据，方法返回outputstream的对象
                oStream  = socket.getOutputStream();
                //3，具体的输出过程
                oStream.write("我是客户端！".getBytes());
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        finally {
                //关闭相应的数据流
                if (oStream != null) {
                        try {
                                oStream.close();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
```

```
        if (socket != null) {
                try {
                        socket.close();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        }

}
@Test
public void  servers1(){
        Socket socket = null;
        ServerSocket  serverSocket=null;
        InputStream iStream = null;
        try {
                //1，创建一个ServerSocket的对象，通过构造器指明自身的端口号
                serverSocket = new ServerSocket(9090);
                //调用其accept（）方法，返回一个socket对象
                socket = serverSocket.accept();
                //调用Socket对象的getInputstream（）获取一个客户端发送过来的输入
流
                iStream  = socket.getInputStream();
                //对应的输入流进行的操作
                byte b[] = new byte[20];
                int len;
                while((len = iStream.read(b)) != -1)
                {
                        String string = new String(b,0,len);
                        System.out.println(string);
                        }
                System.out.println("收到来
自："+socket.getInetAddress().getHostAddress()+"的信息！");
```

```
        }
                catch (Exception e) {
                        // TODO: handle exception
                        e.printStackTrace();
                }
        finally {
                //5，关闭相应的数据流
                if (socket != null) {
                        try {
                                socket.close();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
                if (serverSocket != null) {
                        try {
                                serverSocket.close();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
                if (iStream != null) {
                        try {
                                iStream.close();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
        }


}
```

```
}
```