



```

* InetAddress:位于java.net包下
* 1. InetAddress用来代表IP地址。一个InetAddress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
*   getAddress(): 获取IP地址

```

dns域名解析服务器:

本机IP:

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

其他机器IP:

```
InetAddress inet4Address = (InetAddress)
```

```
InetAddress.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。

TCP/IP协议簇

- 传输层协议中有两个非常重要的协议：
 - 传输控制协议TCP(Transmission Control Protocol)
 - 用户数据报协议UDP(User Datagram Protocol)。
- TCP/IP 以其两个主要协议：**传输控制协议(TCP)**和**网络互联协议(IP)**而得名，实际上是一组协议，包括多个具有不同功能且互为关联的协议。
- IP(Internet Protocol)协议是网络层的主要协议，支持网间互连的数据通信。
- TCP/IP协议模型从更实用的角度出发，形成了高效的四层体系结构，即**物理链路层**、**IP层**、**传输层**和**应用层**。

TCP 和 UDP

TCP协议：

- 使用TCP协议前，须先建立TCP连接，形成传输数据通道
- 传输前，采用“三次握手”方式，是可靠的
- TCP协议进行通信的两个应用进程：客户端、服务端
- 在连接中可进行大数据量的传输
- 传输完毕，需释放已建立的连接，效率低

UDP协议：

- 将数据、源、目的封装成数据包，不需要建立连接
- 每个数据报的大小限制在64K内
- 因无需连接，故是不可靠的
- 发送数据结束时无需释放资源，速度快

Socket

- 利用套接字(Socket)开发网络应用程序早已被广泛的采用，以至于成为事实上的标准。
- 通信的两端都要有Socket，是两台机器间通信的端点
- 网络通信其实就是Socket间的通信。
- Socket允许程序把网络连接当成一个流，数据在两个Socket间通过IO传输。
- 一般主动发起通信的应用程序属**客户端**，等待通信请求的为**服务端**

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();
        }
    }
}

```

```

}

```

