第四章 网络编程（上）

4.1 URL 对象

例：通过 URL 读取 WWW 信息
代码：

```
import java.net.*;
import java.io.*;

public class URLReader{
    public static void main(String[] args) throws Exception{
        URL cs = new URL("http://www.yushuai.me/");
        BufferedReader in = new BufferedReader(new
InputStreamReader(cs.openStream()));
        String inputLine;
        while((inputLine = in.readLine())!=null)
            System.out.println(inputLine);
        in.close();
    }
}
```

构造 URL 对象：
（1）public URL(String spec)

```
URL urlBase = new URL("http://www.yushuai.me");
```

直接把网址作为一个字符串，然后并作为参数传递给 URL 资源。
（2）我还可以用一个 URL 对象来表示它的上下文的这个环境

```
public URL(URL context, String spec)
URL yushuai = new URL("http://www.yushuai.me/pages/1");
URL yush = new URL(yushuai, "Gamelan.game.html");
URL gamelanNetwork = new URL(gamelan, "Gamelan.net.html");
```

获取 URL 对象属性：
public String getProtocol()：返回你所描述的 URL 访问时采用的协议；
public String getHost()：获得它的主机名；
public String getPort()：获得端口号；
public String getFile()：获得文件名；
public String getRef()：获得引用地址。

4.2 URL Connection 对象

这个对象代表了 java 和互联网资源上的一个访问链接，可以通过它对 URL 资源进行读写。它与 URL 的区别是，URL 是单向，而本对象是双向，本对象还可以查看服务器的相响应

消息的首部，它还可以设置客户端请求消息的首部。

1.使用 URLConnection 通信的一般步骤：

（1）构造一个 URL 对象；

（2）调用 URL 对象的 openConnection()方法获得对应 URL 的 URLConnection 对象；

（3）配置此 URLConnect 对象；

（4）读取首部字段；

（5）获得输入流以便进行读取数据；

（6）获得输出流写入数据；

（7）关闭连接。

例：通过 URLConnection 抓取网页

代码：

```
import java.net.*;
import java.io.*;

public class URLReader{
    public static void main(String[] args){
        try {
            URL cs =new URL("http://www.yushuai.me/");
            URLConnection tc =cs.openConnection();
            BufferedReader in = new BufferedReader(new
InputStreamReader(tc.getInputStream()));
            String inputLine;
            while((inputLine = in.readLine())!= null)
                System.out.println(inputLine);
            in.close();
        }catch(MalformedURLException e) {
            System.out.println("MalformedURLException");
        }
        catch(IOException e) {System.out.println("IOException");}
    }
}
```

4.3 Get 请求与 Post 请求

例：发送 get 请求

代码：

```
    public static String sendGet(String url, String param) {
        String result = "";
        BufferedReader in = null;
        try {
            String urlNameString = url + "?" + param;
            URL realUrl = new URL(urlNameString);
            // 打开和URL之间的连接
            URLConnection connection = realUrl.openConnection();
```

```
            // 设置通用的请求属性
            connection.setRequestProperty("accept", "*/*");
            connection.setRequestProperty("connection", "Keep-Alive");
            // 建立实际连接
            connection.connect();
            // 定义BufferedReader输入流来读取URL的响应
            in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String line;
            while ((line = in.readLine()) != null) {
                result += line;
            }
        } catch (Exception e) {
        } finally {
            try {
                if (in != null) {
                    in.close();
                }
            } catch (final Exception e2) {
            }
            return result;
        }
    }
```

发送 POST 请求，POST 请求的参数需要通过 URLConnection 的输出流来写入参数。

例：发送 POST 请求

代码:

```
    public static String sendPost(String url, String param) {
        PrintWriter out = null;
        BufferedReader in =null;
        String result = "";
        try {
            URL realUrl = new URL(url);
            //打開和URL之間的連接
            URLConnection conn = realUrl.openConnection();
            //设置通用的请求属性
            conn.setRequestProperty("accept", "*/*");
            conn.setRequestProperty("connection", "Keep-Alive");
            //允许输出流
            conn.setDoOutput(true);
            //获取URLConnection对象对应的输出流
            out = new PrintWriter(conn.getOutputStream());
            //发送请求参数
            out.print(param);
            out.flush();
```

```
            in = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
            String line;
            while((line = in.readLine())!=null) {
                result +=line;
            }
        }
        catch(Exception e) {}
        finally {
            try {
                if(out!=null) {
                    out.close();
                }
                if(in!=null) {
                    in.close();
                }
            }catch(IOException ex) {}
        }
        return result;
    }
```

HttpURLConnection 类在 URLConnection 的基础上提供一系列针对 http 请求的内容:

（1）HTTP 状态码（例如 HTTP_OK:200）；

（2）setRequestMethod（设置请求方法 get/post 等）;

（3）getResponseCode（获取 HTTP 响应）。

4.4 Socket 通信原理

本节讲述如何在两个 java 程序之间建立网络连接。

网络上的两个程序通过一个双向的通讯连接实现数据的交换，这个双向链路的一端成为一个 socket。Socket 通常用来实现客户方和服务方的连接。

怎样用 socket 大道客户端和服务端之间的通信呢?

如图 1 所示，客户端有一个客户端的 ClientSocket，在 Server 端也有一个 ServerStocket。由客户端向服务器端发出一个 Socket 连接请求，服务器收到后会给它返回一个响应信号，在客户端就建立了连接。

4.5 Socket 通信实现

1.创建 Socket

①Socket()

②Socket(InetAddress address, int port);

③Socket(String host, int port);

④Socket(InetAddress host, int port, InetAddress localAddr, int localPort);

⑤Socket(String host, int port, InetAddress localAddr, int localPort)

例：客户端 Socket 的建立

代码：

```
try{
    Socket socket = new Socket("127.0.0.1", 2000);//2000 是端口号
}catch(IOException e){
    System.out.println("Error:" +e);
}
```

例：服务器端 Socket 的建立

代码：

```
ServerSocket server =null;
try{
    server = new ServerSocket(2000);
}catch(IOException e){
    System.out.println("can not listen to:" +e);
}
Socket socket = null;
try{
    socket = server.accept();
//accept 方法代表等着客户端发送链接请求，如果没有客户端发请求，它就一致循环执行，直到客户端发送请求来。
}catch(IOException e){
    System.out.println("Error:" +e);
}
```

2.打开输入/输出流

利用 socket 构建几个输入输出流。如图 2 所示，第一行就是通过我们一个 socket 对象的 getoutputstream 方法获得一个输出流，反过来理解就是拿到 socket 输入流以后外面包一层 BufferedOutputStream，然后外面再包一层 PrintStream，这样利用 PrintStream 就可以进行通讯了。只不过这是一个输出流。输出流是干什么用的呢？站在 java 程序来看，这个输出流就是往外发送信息的。

第二行来说，这是一个输入流。我们通过 socket 对象的 getInputStream 方法获得了输入流，然后在外面包了一层 DataInputStream。

所以我们构造了这个 socket 输入输出流了以后，就可以用来进行通讯。第三行和第四行是另外一种方式来构造 socket 的输入输出流。

3.进行读写操作。

4.关闭 socket

在关闭 socket 之前，先关闭输入输出流，使用 close()方法。记住，**先关闭流，再关闭 socket。**

例：命令行聊天程序

客户端代码：

```
import java.io.*;
import java.net.*;

public class TalkClient {
```

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            //构造一个socket，端口号4700
            Socket socket = new Socket("127.0.0.1", 4700);
            //构造三个输入输出流。第一个sin是键盘输入流。第二个os输出流，从客户端发信息给服务端
            BufferedReader sin = new BufferedReader(new InputStreamReader(System.in));
            PrintWriter os = new PrintWriter(socket.getOutputStream());
            //第三个是接受网络上发给我的信息
            BufferedReader is = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            //接受信息
            String readline;
            readline = sin.readLine();
            while(!readline.equals("bye"))
            {
                os.println(readline);
                os.flush();
                //下面是打印客户端和服务器端发送的信息
                System.out.println("Client:"+ readline);
                System.out.println("Server:"+ is.readLine());
                readline = sin.readLine();//再获取客户键盘输入
            }
            os.close();
            is.close();
            socket.close();//先关闭流再关闭socket
        }catch(Exception e) {
            System.out.println("Error"+e);
        }

    }

}
```

服务器端代码：

```
import java.io.*;
import java.net.*;
import java.applet.Applet;

public class TalkServer {
```

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            ServerSocket server = null;//声明一个serversocket对象
            try {
                server = new ServerSocket(4700);
            }catch(Exception e) {
                System.out.println("can not listen to:" +e);
            }
            Socket socket = null;
            try {
                socket = server.accept();
            }catch(Exception e) {
                System.out.println("Error."+e);
            }
            //下面开始通信。第一个流是读入客户端发送的信息。
            //第二个是发送信息给客户端，第三个是键盘输入流
            String line;
            BufferedReader is = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter os =new PrintWriter(socket.getOutputStream());
            BufferedReader sin = new BufferedReader(new
InputStreamReader(System.in));
            System.out.println("Client:"+is.readLine());
            line =sin.readLine();
            while(!line.equals("bye"))
            {
                os.println(line);
                os.flush();
                //下面是打印客户端和服务器端发送的信息
                System.out.println("Server:"+ line);
                System.out.println("Client:"+ is.readLine());
                line = sin.readLine();

            }
            os.close();
            is.close();
            server.close();
        }catch(Exception e) {\
            System.out.println("Error:"+e);
        }

    }
```

```
}
```