به نام خدا

برنامه‌سازی پیشرفته

دانشگاه شهید بهشتی · دانشکده مهندسی و علوم کامپیوتر

دکتر مجتبی وحیدی اصل

برنامه‌نویسی شبکه - بخش دوم

قائم علی‌آبادی

Java URL

- کلاس `java.net.URL` نماینده یک **URL** است.
- **URL** در واقع Uniform Resource Locator است و به یک منبع در وب (یا روی پروتکل‌های دیگر) اشاره می‌کند.
- نمونه:

```
https://www.javazava.com/java-tutorial
```

اجزای اصلی در یک URL

- **Protocol**: در مثال بالا `https` پروتکل است.
- **Host name / IP**: در مثال بالا `www.javazava.com` نام میزبان است (می‌تواند IP هم باشد).
- **Port number**: ویژگی اختیاری؛ اگر بنویسیم `http://www.javazava.com:80/` پورت `80` است. اگر پورت ذکر نشود، متد `getPort()` عدد `-1` برمی‌گرداند.
- **File/Directory**: مسیر یا نام فایل، در مثال بالا `java-tutorial` است.

متدهای پرکاربرد کلاس Java URL

Method	توضیح
`public String getProtocol()`	پروتکلِ URL را برمی‌گرداند (مثل `http` یا `https`).
`public String getHost()`	نام میزبان یا IP را برمی‌گرداند.
`public int getPort()`	شماره پورت URL را برمی‌گرداند؛ اگر تعیین نشده باشد مقدار `-1`.
`public String getFile()`	نام فایل/مسیر انتهایی URL را برمی‌گرداند.
`public String getAuthority()`	بخش Authority (معمولا `host:port`) را برمی‌گرداند.
`public String toString()`	بازنمایی متنی کاملِ URL را برمی‌گرداند.
`public String getQuery()`	رشته Query (پس از `?`) را برمی‌گرداند.
`public int getDefaultPort()`	پورت پیش‌فرض پروتکل را برمی‌گرداند (مثلا 80 برای HTTP).
`public URLConnection openConnection()`	یک شئ `URLConnection` مرتبط با این URL ایجاد می‌کند.
`public boolean equals(Object obj)`	برابری URL فعلی با شئ داده‌شده را مقایسه می‌کند.
`public Object getContent()`	محتوای URL را (بسته به نوع منبع) برمی‌گرداند.
`public String getRef()`	Fragment/Anchor (پس از `#`) را برمی‌گرداند.
`public URI toURI()`	نمایش `URI` متناظر با URL را بازمی‌گرداند.

```java
import java.net.*;

public class URLDemo {
    public static void main(String[] args) {
        try {
            URL url = new URL("https://www.google.com/search?q=java&ie=&oe=");
            System.out.println("Protocol: " + url.getProtocol());
            System.out.println("Host Name: " + url.getHost());
            System.out.println("Port Number: " + url.getPort());
            System.out.println("Default Port Number: " + url.getDefaultPort());
            System.out.println("Query String: " + url.getQuery());
            System.out.println("Path: " + url.getPath());
            System.out.println("File: " + url.getFile());
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
Protocol: https
Host Name: www.google.com
Port Number: -1
Default Port Number: 443
Query String: q=java&ie=&oe=
Path: /search
File: /search?q=java&ie=&oe=
```

کلاس Java URLConnection

- کلاس `java.net.URLConnection` یک **پیوند ارتباطی** بین برنامه و منبعی است که URL به آن اشاره می‌کند.
- از این کلاس می‌توان برای **خواندن** و **نوشتن** داده روی منبع مشخص‌شده استفاده کرد.
- **دریافت شیء URLConnection**: متد `openConnection()` در کلاس `URL` شیء `URLConnection` را برمی‌گرداند.

```
public URLConnection openConnection() throws IOException { }
```

- **نمایش کد منبع یک صفحه وب**: با گرفتن `InputStream` از شیء اتصال (`getInputStream()`) می‌توان داده‌های صفحه را خواند و چاپ کرد.

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

public class URLConExample {
    public static void main(String[] args) {
        try {
            URL url = new URL("https://sbu.ac.ir/");
            URLConnection urlcon = url.openConnection();
            InputStream stream = urlcon.getInputStream();
            int i;
            while ((i = stream.read()) != -1) {
                System.out.print((char) i);
            }
            stream.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

کلاس Java HttpURLConnection

- `java.net.HttpURLConnection` زیرکلاس `URLConnection` و مخصوص پروتکل‌های **HTTP/HTTPS** است.
- با این کلاس می‌توان اطلاعات HTTP مانند **Header**ها، **Status/Response Code**، بدنه پاسخ و حتی ارسال درخواست‌های POST/GET را مدیریت کرد.
- **دریافت شیء**: متد `openConnection()` روی شیء `URL` اجرا می‌شود و می‌توان خروجی‌اش را به `HttpURLConnection` **Cast** کرد.

```
URL url = new URL("http://www.java.com/java-tutorial");
HttpURLConnection huc = (HttpURLConnection) url.openConnection();
```

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

public class HTTPURL {
    public static void main(String[] args) {
        try {
            URL url = new URL("https://sbu.ac.ir/");
            HttpURLConnection huc = (HttpURLConnection) url.openConnection();
            for (int i = 1; i <= 8; i++) {
                System.out.println(huc.getHeaderFieldKey(i) + " = " + huc.getHeaderField(i));
            }
            huc.disconnect();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

کلاس Java InetAddress

- کلاس `java.net.InetAddress` نماینده یک آدرس IP است.
- این کلاس متدهایی برای به‌دست‌آوردن IP از روی نام میزبان (مثل `www.yahoo.com`، `www.google.com`) و برعکس در اختیار می‌گذارد.
- آدرس IP به‌صورت 32بیتی (IPv4) یا 128بیتی (IPv6) نمایش داده می‌شود؛ هر شئ `InetAddress` ترکیب «IP + نام میزبان» را نمایش می‌دهد.
- نمونه‌های این کلاس معمولا برای کار با کلاس‌های شبکه مانند `Socket`، `ServerSocket`، `DatagramPacket` و `DatagramSocket` استفاده می‌شوند.
- این کلاس سازنده عمومی ندارد؛ ایجاد شیء از طریق **factory methods** انجام می‌شود.

متدهای پرکاربرد

Method	توضیح
`public static InetAddress getByName(String host) throws UnknownHostException`	نمونه‌ای از `InetAddress` برای نام میزبان داده‌شده برمی‌گرداند (نگاشت نام→IP).
`public static InetAddress getLocalHost() throws UnknownHostException`	نمونه‌ای از `InetAddress` مربوط به میزبان محلی (نام و IP دستگاه جاری) را بازمی‌گرداند.
`public String getHostName()`	نام میزبان مرتبط با این آدرس IP را برمی‌گرداند.
`public String getHostAddress()`	رشته متنی آدرس IP را برمی‌گرداند.

```java
import java.net.*;

public class INETDemo {
    public static void main(String[] args) throws Exception {
        InetAddress address1 = InetAddress.getByName("www.yahoo.net");
        System.out.println(address1.getHostAddress() + " " + address1.getHostName());
        System.out.println(address1);

        InetAddress address2 = InetAddress.getLocalHost();
        System.out.println(address2);
    }
}
```

```
76.223.84.192 www.yahoo.net
www.yahoo.net/76.223.84.192
Arass-MacBook-Pro.local/127.0.0.1
```

DatagramPacket و Java DatagramSocket

- کلاس‌های `DatagramSocket` و `DatagramPacket` برای برنامه‌نویسی سوکت **بدون اتصال (UDP)** استفاده می‌شوند.

DatagramSocket

- نمایانگر یک سوکت اتصال‌ناپذیر برای **ارسال/دریافت بسته‌های دیتاگرام** است.
- **Datagram** صرفا حاوی داده است و هیچ تضمینی برای صحت محتوا یا زمان رسیدن ندارد.

سازنده‌های پرکاربرد:

- `DatagramSocket()` : ایجاد سوکت و bind روی یک پورت در دسترس روی localhost.
- `DatagramSocket(int port)` : ایجاد سوکت و bind روی پورت مشخص.
- `DatagramSocket(int port, InetAddress address)` : ایجاد سوکت و bind روی پورت و IP محلی مشخص.

DatagramPacket

سازنده‌های پرکاربرد:

- `DatagramPacket(byte[] buf, int length)` : دیتاگرام برای **دریافت** بسته‌ها.
- `DatagramPacket(byte[] buf, int length, InetAddress address, int port)` : دیتاگرام برای **ارسال** بسته‌ها.

```java
// Sender (DSender.java)
import java.net.*;

public class DSender {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        String str = "Welcome java";
        InetAddress ip = InetAddress.getByName("127.0.0.1");
        DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);
        ds.send(dp);
        ds.close();
    }
}
```

```java
// Receiver (DReceiver.java)
import java.net.*;

public class DReceiver {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(3000);
        byte[] buf = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, 1024);
        ds.receive(dp);
        String str = new String(dp.getData(), 0, dp.getLength());
        System.out.println(str);
        ds.close();
    }
}
```

ارتباط بدون اتصال (UDP) - سرور/کلاینت

- برای ارسال یک بسته با UDP باید چهار چیز را بدانیم: **پیام**، **طول پیام**، **آدرس IP مقصد** و **شماره پورت مقصد**.
- بعد از مشخص شدن این موارد، یک شیء `DatagramSocket` و سپس `DatagramPacket` می‌سازیم تا داده را حمل کند.
- با متدهای `send()` / `receive()` بسته‌ها واقعا ارسال/دریافت می‌شوند.
- پس از دریافت، داده داخل بسته را استخراج می‌کنیم (مثلا با ساختن `String` از بایت‌ها).

```java
// UDP Client - sends console input to server (port 1234) until "bye"
import java.io.*;
import java.net.*;
import java.util.*;

public class udpBaseClient_2 {
    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        DatagramSocket ds = new DatagramSocket();
        InetAddress ip = InetAddress.getLocalHost();
        byte[] buf;
        while (true) {
            String inp = sc.nextLine();
            buf = inp.getBytes();
            DatagramPacket dpSend = new DatagramPacket(buf, buf.length, ip, 1234);
            ds.send(dpSend);
            if (inp.equals("bye")) break;
        }
        ds.close();
    }
}
```

```java
// UDP Server - listens on port 1234 and echoes received bytes; exits on "bye"
import java.io.*;
import java.net.*;

public class udpBaseServer_2 {
    public static void main(String[] args) throws IOException {
        DatagramSocket ds = new DatagramSocket(1234);
        byte[] receive = new byte[65535];
        DatagramPacket dpReceive;
        while (true) {
            dpReceive = new DatagramPacket(receive, receive.length);
            ds.receive(dpReceive);
            StringBuilder sb = data(receive);
            System.out.println("Client:- " + sb);
            if (sb.toString().equals("bye")) {
                System.out.println("Client sent bye.....EXITING");
                break;
            }
            receive = new byte[65535];
        }
        ds.close();
    }

    public static StringBuilder data(byte[] a) {
        if (a == null) return null;
        StringBuilder ret = new StringBuilder();
        int i = 0;
        while (a[i] != 0) {
            ret.append((char) a[i]);
            i++;
        }
        return ret;
    }
}
```

راهنمای اجرای نمونه UDP (Client-Server)

- برای تست برنامه‌ها، ابتدا **سرور** را اجرا کنید، سپس **کلاینت** را.
- در کنسول کلاینت، پیام‌های خود را تایپ و ارسال کنید (هر خط یک پیام).
- برای خاتمه ارتباط، عبارت `bye` را ارسال کنید.
- تمرین: نسخه دوطرفه گفت‌وگو (chat) را پیاده‌سازی کنید تا سرور هم بتواند در هر زمان پاسخ دهد.

خودمون رو بسنجیم

این بخش برای این طراحی شده که در پایان مطالعه این اسلاید، بتونی خودت رو محک بزنی و ببینی آیا مفاهیم رو به خوبی یاد گرفتی یا نه. سوالات زیر رو مرور کن و سعی کن بدون نگاه کردن به متن درس، به اون ها پاسخ بدی.

- یک URL رو به اجزای اصلی (پروتکل، میزبان، پورت، مسیر) تجزیه کن.
- دو متد از `URL` که برای استخراج اطلاعات URL استفاده می‌شوند رو نام ببر و مثال بزن.
- برای ارسال یک دیتاگرام UDP چه اطلاعاتی لازمه و کدوم کلاس‌ها رو استفاده می‌کنیم؟
- در سرور UDP چرا بافر رو بعد از هر دریافت دوباره مقداردهی می‌کنیم؟
- تفاوت رویکرد blocking در TCP مثال‌ها با رفتار connection-less در UDP چیست؟

پایان

اگه سوال و پیشنهادی داشتید می‌تونید از هر دو طریق زیر با من در ارتباط باشید. :)

Email: me@ghaem.me
Telegram: @Ghaem