InetAddress类与InetSocketAddress类

1. **InetAddress类**：存在于**java.net包**中，有两个直接子类，分别是Inet4Address, Inet6Address，直接继承于java.lang.Object类，Inet4Address和Inet6Address的方法大多是从**InetAddress**继承下来的。此类用来标识互联网的IP地址。
2. **InetAddress类没有构造方法，利用静态方法获取实例对象**。

静态方法：可以通过**域名或IP地址**获取Internet上主机的IP地址对象。通过静态方法getLocalHost可以获取本地主机的地址。

1. getByName（String host）：**通过字符串**，可以是**域名或者IP地址**。



1. getByAddress（byte[] addr）**IPv4为4个字节，IPv6为16个字节**。

把IP地址封装到字节数组中，对于大于127的数据，需要转成byte类型。

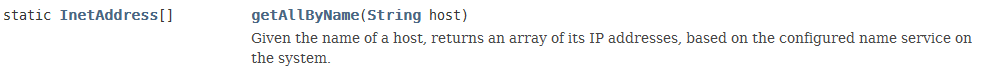


1. getByAddress（String host，byte[] addr）



1. **getAllByName（String host）：返回InetAddress对象数组。**

**既可以解析域名，也可以解析ip地址的字符串形式。**



如：System.out.println(InetAddress.getByName("www.baidu.com").getHostName());//通过域名获取的地址对象，名字就是域名。

System.out.println(InetAddress.getByName("180.149.132.151").getHostName());//通过ip地址获取的地址对象，名字就是ip地址。这是为了安全起见，防止DNS没有成功解析此ip地址而出现异常。



1. getLocalHost（）：**获得本地主机的IP地址对象。**

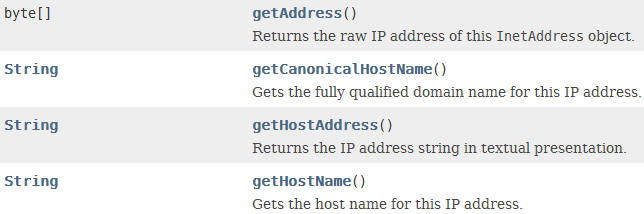


1. getLoopbackAddress（）：**获得回送IP地址对象**。



注: loopback [lup'bæk] n. [计] 回送；回路

1. 实例方法：
2. getXxx方法：

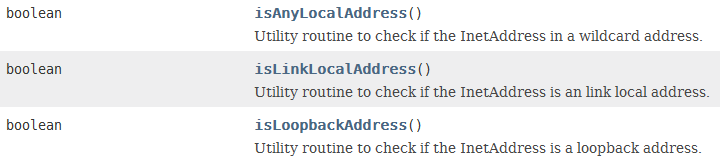


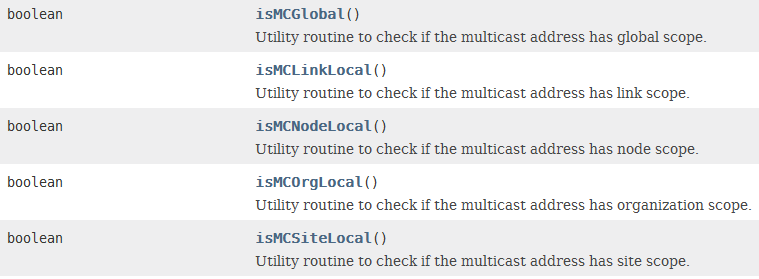
1. equals方法、toString()、hashCode（）方法。

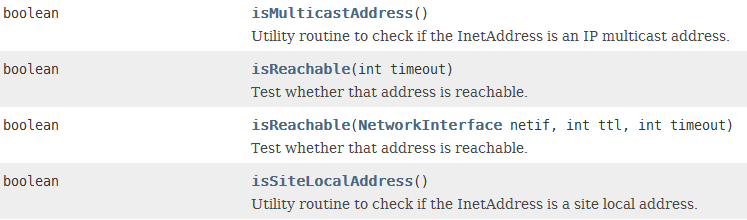
InetAddress的toString方法的输出语句结构：

www.tsinghua.edu.cn/166.111.4.100

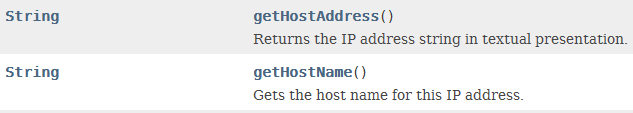
1. isXxx判断方法:







1. 其他继承Object的方法。
2. 注意**域名与IP地址**的映射：



如果存在域名或DNS进行了解析，则getHostName返回域名，否则直接返回IP地址。



1. **InetAddress类的创建对象方法**的练习:

public class InetAdressDemo {

public static void main(String[] args) throws UnknownHostException {

**// InetAddress ipd = InetAddress.getByName("www.tsinghua.edu.cn");**

String strip = new String("166.111.4.100");

**// InetAddress ipd = InetAddress.getByName(strip);//与上面的等价**

String[] ips = strip.split("\\.");

byte[] bs = new byte[4];

for (int i = 0; i < ips.length; i++) {

bs[i]=(byte)Integer.parseInt(ips[i]);

}

**// InetAddress ipd = InetAddress.getByAddress(bs);//**

InetAddress ipd = **InetAddress.getByAddress("www.tsinghua.edu.cn",bs);//**

**//InetAddress ipd = InetAddress.getLocalHost();//获取本地主机地址对象**

String address = ipd.getHostAddress();

String name = ipd.getHostName();

byte[] ip = ipd.getAddress();

System.out.println(address + " "+name);

StringBuilder sb = new StringBuilder();

for(byte b : ip) {

sb.append(Byte.toUnsignedInt(b)+".");

}

sb.delete(sb.length()-1,sb.length());

System.out.println(sb);

}

1. SocketAddress类也存在于java.net 包中，直接继承于java.lang.Object包中，有一个直接子类**InetSocketAddress类**。



注意:InetAddress只封装了IP地址和域名，而InetSocketAddress不仅封装了IP地址和域名，还封装了端口号。

1. **SocketAddress类**只有一个空参数的构造方法，没有自己独有的成员方法。

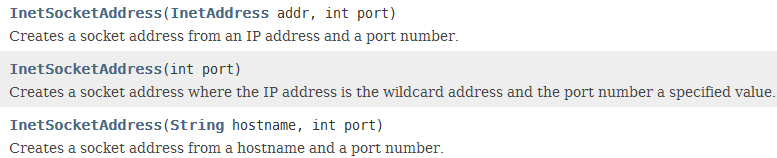


1. 使用的主要是其子类**InetSocketAddress**，也存在于java.net包中，直接继承于SocketAddress类：

套接字地址：**IP地址+端口号**或者**域名+端口号**。**本质就是把一个InetAddress对象和一个int值封装在一起。**

1. InetSocketAddress的构造方法：

**InetSocketAddress = InetAddress+port。**

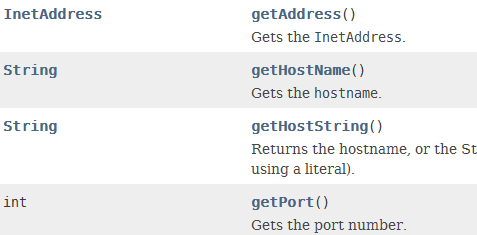


1. InetSocketAddress的静态方法：



1. **InetSocketAddress的成员方法**：
2. getXxx方法：

很显然会有getAddress方法，返回一个InetAddress对象，还会有getPort方法，返回一个int值，即端口号。getHostName可以返回主机名称。



1. 判断方法：



1. equals、hashCode、toString方法:



1. URL与URI、URN：

**URI，是uniform resource identifier，统一资源标识符**，用来唯一的标识一个资源。而**URL是uniform resource locator，统一资源定位器，它是一种具体的URI，**即URL可以用来标识一个资源，而且还指明了如何locate这个资源。而**URN，uniform resource name，统一资源命名**，是通过名字来标识资源，比如mailto:java-net@java.sun.com。也就是说，URI是以一种抽象的，高层次概念定义统一资源标识，而URL和URN则是具体的资源标识的方式。URL和URN都是一种URI。

总结一下：**URL是一种具体的URI，它不仅唯一标识资源，而且还提供了定位该资源的信息。URI是一种语义上的抽象概念，可以是绝对的，也可以是相对的，而URL则必须提供足够的信息来定位，所以，是绝对的，而通常说的relative URL，则是针对另一个absolute URL，本质上还是绝对的。**

注：这里的绝对(absolute)是指包含scheme，而相对(relative)则不包含scheme。

1. **URL类**存在于**java.net包**中，直接继承于**java.lang.Object类**。
2. 一个URL对象通常包括最基本的三部分信息：

**协议、地址、端口号、资源。**

（1）**协议**常用的是HTTP、FTP、FILE协议，这三个都是Java虚拟机都支持的协议。

（2）**地址**是IP地址或域名。

（3）**端口号**：

（4）**资源**可以是任意一个文件。

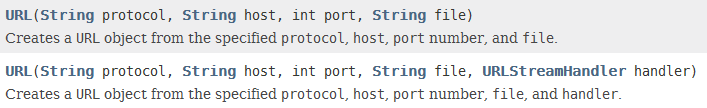
1. **预想读取URL中的资源：需要利用成员方法openStream（）方法获取一个输入流。**
2. **URL类的构造方法**：这些构造方法都可能会抛出[MalformedURLException](mk:@MSITStore:E:\JAVA\JAVA%20JIHE\API帮助文档\CHM帮助文档\jdk-8u60-docs-all_API.chm::/api/java/net/MalformedURLException.html)异常。
3. 利用**字符串**初始化一个URL对象：



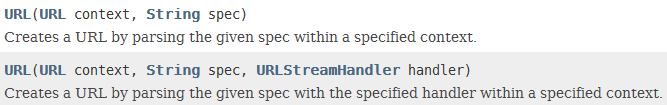
1. **利用协议、地址、资源文件定义一个URL对象：（端口号是默认的）**



1. **利用协议、地址、端口号、资源文件定义一个URL对象：**



1. 利用一个URL：



1. **URL类只有一个静态方法：**

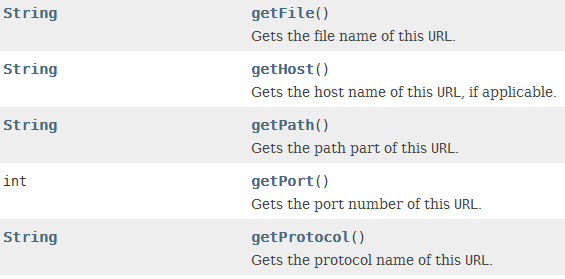


1. **URL类的成员方法：**
2. 获取方法getXxxx（）

①**既然URL包括四部分，即四个组成要素，对应的有具体的方法获取**：**获取的都是字符串形式。**

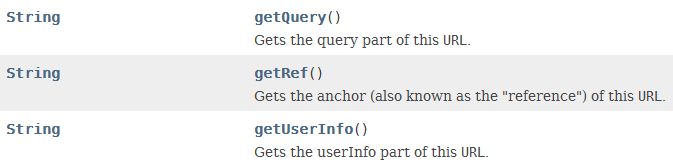
getProtocol（）、getHost（）、getPort（）、getDefaultPort（）方法、getFile（），

getPath（）：获取资源的相对路径。





②getQuery（）：获取锚点对应的值。getRef（）、getUserInfo（）。



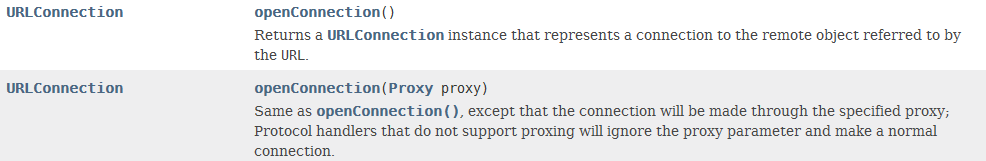
③getAuthority（）、getContent（）。



1. 覆写的equals（）、hashCode（）方法。
2. **读取URL中的资源**，**通过openStream（）方法返回一个字节输入流。**



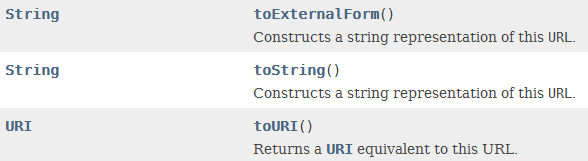
1. openConnection（）方法：



1. 判断方法**：sameFile（）：是不是同一个资源。**



1. toXxx（）方法：



1. URL类的示例：

public class URLDemo {

public static void main(String[] args) throws IOException {

URL url = null;

InputStream is = null;

OutputStream os = null;

url = new URL("http://www.tsinghua.edu.cn/publish/newthu/index.html");

String host = url.getHost();

String file = url.getFile();

String protocol = url.getProtocol();

int port = url.getPort();

int defaultPort = url.getDefaultPort();

System.out.println(host + "\n" + file + "\n" + protocol + "\n" + port

+ " " + defaultPort);

**is = url.openStream();**

**os = new FileOutputStream("e:\\11.txt");**

**byte[] buf = new byte[1024];**

**int len = 0;**

**while ((len = is.read(buf)) != -1) {**

**os.write(buf, 0, len);**

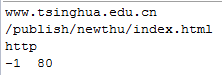
**}**

is.close();

os.close();

}

结果：



1. **注意：由于不同的平台的编码表不一样，所以有时候需要利用转换流，InputStreamReader和OutputStreamWriter。**