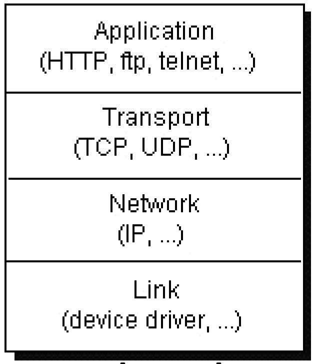
Java.net Package and Classes(Url, UrlConnection, HttpUrlConnection)

Sisoft Technologies Pvt Ltd
SRC E7, Shipra Riviera Bazar, Gyan Khand-3, Indirapuram, Ghaziabad
Website: www.sisoft.in Email:info@sisoft.in
Phone: +91-9999-283-283

Java Networking



- Networking is a concept of connecting two or more computing devices together so that resource can be shared.
- Java programs that communicate over network usually work in Application layer
- The java.net package provides support for the two common network protocols
  - TCP: *TCP* (*Transmission Control Protocol*) is a connection-based protocol that provides a reliable flow of data between two computers. The Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Telnet are all examples of applications that require a reliable communication channel.
  - UDP: *UDP* (*User Datagram Protocol*) is a protocol that sends independent packets of data, called datagrams, from one computer to another with no guarantees about arrival. UDP is not connection-based like TCP.

Java.net package

- The URL, URLConnection, Socket, and ServerSocket classes all use TCP to communicate over the network
- The DatagramPacket, DatagramSocket, and MulticastSocket classes are for use with UDP
- URL: A URL (Uniform Resource Locator) is the address of a resource on the Internet. Java programs can use URLs to connect to and retrieve information over a network. This lesson will provide a more complete definition of a URL and shows you how to create and parse a URL, how to open a connection to a URL, and how to read from and write to that connection.
- Sockets:A socket is one endpoint of a two-way communication link between two programs running on the network. This is used for client/server communication eg. a client can connect to a standard server, the Echo server, and communicate with it via a socket

URL, URLConnection HTTPURL Connection

URL Object

- URL is an acronym for *Uniform Resource Locator* and is a reference (an address) to a resource on the Internet.
- A URL has two main components:
  - Protocol identifier: For the URL http://sisoft.in, the protocol identifier is http.
  - Resource name: For the URL http://sisoft.in, the resource name is sisoft.in.
- URL object may be created by passing string URI (Uniform Resource Identifier) to URL Constructor
  - URL url = **new** URL(**str_url**);
- This string uri may be built by concatenating different arguments
- The multi-argument constructors of the java.net.URI class may also be used. One of the constructors is as follow:
- URI(String scheme, String host, String path, String fragment)
- And then convert the URI to a URL.
- URL url = uri.toURL();

URLConnection

- Call the URL object's openConnection method to get a URLConnection object, or one of its protocol specific subclasses, e.g. java.net.HttpURLConnection
- This URLConnection object to setup parameters and general request properties that you may need before connecting. Connection to the remote object represented by the URL is only initiated when the URLConnection.connect method is called.
- You are not always required to explicitly call the connect method to initiate the connection. Operations that depend on being connected, like getInputStream, getOutputStream, etc, will implicitly perform the connection, if necessary.

Displaying source code of a webpage by URLConnecton class

```
import java.io.*;
import java.net.*;
public class URLConnectionExample
 {
public static void main(String[] args)
{
try{
URL url=new URL("http://www.sisoft.in");
URLConnection urlcon=url.openConnection();
InputStream stream=urlcon.getInputStream();
int i;
while((i=stream.read())!= -1){
System.out.print((char)i);
}
}catch(Exception e){System.out.println(e);}
}
}
```

Java HttpURLConnection class

- The java.net.HttpURLConnection is subclass of URLConnection class
- The **Java HttpURLConnection** class is http specific URLConnection. It works for HTTP protocol only.
- By the help of HttpURLConnection class, you can information of any HTTP URL such as header information, status code, response code etc.
- **How to get the object of HttpURLConnection class**

  The openConnection() method of URL class returns the object of URLConnection class.

  Syntax:

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

HttpURLConnection – Request Buffer Handling

- Set Request method for Connection using methods
  - urlConn.setRequestMethod(“GET”)   // default is GET
  - urlConn.setRequestMethod(“POST”) ;
- In case of Post method, parameter must be sent along with Request Buffer. For this, URLConnection must open output stream.
  - urlConn.setOutput(true)
- Format the post parameters in string in key=value pair connected by “&” sign
  - String postParameters = “id=302&num=1234567"
- Open output stream and preferable change this to BufferedOutputStream and write the post parameter on this outputStream
- The output stream must be flushed and closed.

Example:

```
      public class HttpURLConnectionExample
{
private static final String USER_AGENT = "Mozilla/5.0";
 private static final String GET_URL =http://localhost:9090/SisoftExample";
private static final String POST_URL =
      "http://localhost:9090/SisoftExample/home";

private static final String POST_PARAMS = "userName=Pankaj";
public static void main(String[] args) throws IOException
 {
 sendGET();
System.out.println("GET DONE");
sendPOST();
System.out.println("POST DONE");
 }
```

```
private static void sendGET() throws IOException
{
URL obj = new URL(GET_URL);
 HttpURLConnection con = (HttpURLConnection) obj.openConnection();
 con.setRequestMethod("GET");
 con.setRequestProperty("User-Agent", USER_AGENT);
 int responseCode = con.getResponseCode();
System.out.println("GET Response Code :: " + responseCode);
if (responseCode == HttpURLConnection.HTTP_OK)
{
 // success
 BufferedReader in = new BufferedReader(new InputStreamReader(
     con.getInputStream()));
String inputLine; StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null)
{
 response.append(inputLine);
}
 in.close();
// print result
System.out.println(response.toString());
}
 else
{
 System.out.println("GET request not worked");
 }
 }
```

```
private static void sendPOST() throws IOException
{
URL obj = new URL(POST_URL);
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
 con.setRequestMethod("POST");
 con.setRequestProperty("User-Agent", USER_AGENT);
// For POST only - START
con.setDoOutput(true);
OutputStream os = con.getOutputStream();
 os.write(POST_PARAMS.getBytes());
os.flush();
os.close();
// For POST only - END
int responseCode = con.getResponseCode();
 System.out.println("POST Response Code :: " + responseCode);
if (responseCode == HttpURLConnection.HTTP_OK) //success
{
BufferedReader in = new BufferedReader(new InputStreamReader(
      con.getInputStream()));
String inputLine;
 StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
 response.append(inputLine);
 }
in.close();
 System.out.println(response.toString()); // print result
 }
 else  {
 System.out.println("POST request not worked");
} } }
```

Output:

```
GET Response Code :: 200 <html><head>
<title>Home</title></head><body><h1> Hello world! </h1><P> The time on
the server is March 6, 2015 9:31:04 PM IST. </p></body></html> GET DONE
POST Response Code :: 200 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML
4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd"><html><head><meta http-
equiv="Content-Type" content="text/html; charset=UTF-8"><title>User
Home Page</title></head><body><h3>Hi Pankaj</h3></body></html>
POST DONE
```