Chapter -3

URLs and URIs

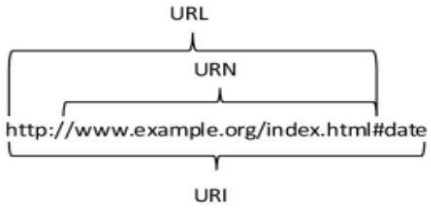
(5 Hours)

Contents

- **URIs**: URLs and Relative URLs
- **The URL Class:** Creating New URLs, Retrieving Data From a URL, Splitting a URL into Pieces, Equality & Comparison and Conversion
- **The URI Class**: Constructing a URI, The Parts of the URI, Resolving Relative URIs, Equality & Comparison and String Representation
- **x-www-form-urlencoded:** URL Encoder and URL Decoder
- **Proxies**: System Properties, The ProxyClass and The ProxySelector Class
- Communicating with Server-Side Programs Through GET
- **Accessing Password-Protected Sites**: The Authenticator Class, The Password Authentication Class and The JPasswordField Class

URLs

- The **URL** class is the simplest way for a Java program to locate and retrieve data from the network.
- We do not need to worry about the details of the protocol being used, the format of the data being retrieved, or how to communicate with the server; simply tell Java the URL and it gets the data



URL Class

BCA 6th

- URL Class represents a Uniform Resource Locator, a pointer to a "resource" on the world wide web.
- **Absolute URL**: contains all of the information necessary to reach the resource.

E.g. /about

- **Relative URL**: contains only enough information to reach the resource relative to (or in the context) another URL.

E.g. http://www.example.com/about

Components of URL

BCA 6th

- **Protocol:** HTTP is the protocol here
- **Hostname:** Name of the machine on which the resource lives.
- **File Name:** The path name to the file on the machine.
- **Port Number:** Port number to which to connect (typically optional).

BCA 6th

http://www.ambition.edu.np/notices/entrance.pdf

- **Base URL**

http://www.ambition.edu.np/notices/

- **Relative URL String**

entrance.pdf

- **Resolved URL**

http://www.ambition.edu.np/notices/entrance.pdf

In JAVA

BCA 6th

```
import java.net.*;

String baseURLStr = "http://www.ietf.org/rfc/rfc3986.txt";
    String relativeURLStr = "rfc2732.txt";
    URL baseURL = new URL(baseURLStr);
    URL resolvedRelativeURL = new URL(baseURL, relativeURLStr);
    System.out.println("Base URL:" + baseURL);
    System.out.println("Relative URL  String:" + relativeURLStr);
    System.out.println("Resolved Relative URL:" + resolvedRelativeURL);
```

Creating URL

BCA 6th

(a) creates a URL with string url representation

- URL url1 = new URL("https://www.google.com/search?q=computer+engineer&sclient=gws-wiz");

(b) creates a URL with a protocol, hostname,and path

- URL url2 = new URL("http", "www.google.com","/contact/");

(b) creates a URL with a protocol, hostname, port and path

- URL url2 = new URL("http", "www.google.com",8008,"/contact/");

(b) creates a URL with a url and string relative

- URL u1 = new URL("http://www.ibiblio.org/javafaq/index.html");
- URL u2 = new URL (u1, "mailinglists.html");

Constructing a URL from a string

BCA 6th

Which protocols does a virtual machine support?

```
try {
        URL u1 = new URL("http://www.ambition.edu.np/");
        System.out.println(u1.getProtocol()); // http
        URL u2 = new URL("verbatim:http://www.adc.org/");
        System.out.println(u2.getProtocol()); // error
    }
     catch (MalformedURLException ex) {
        System.err.println(ex);
}
```

Methods that retrieve data from a URL

BCA 6th

- public InputStream openStream(): connects to the resource referenced by the URL, performs any necessary handshaking between the client and the server, and returns an Input Stream from which data can be read.
- public URLConnection openConnection(): opens a socket to the specified URL and returns a URLConnection object.
- public Object getContent(): method retrieves the data referenced by the URL and tries to make it into some type of object.

openStream()

BCA 6th

- public final InputStream openStream() throws IOException

```
try {
    URL u = new URL("http://www.tufohss.edu.np");
    InputStream in = u.openStream();
    int c;
    while ((c = in.read()) != -1)
        System.out.write(c);
    in.close();
} catch (IOException ex) {
        System.err.println(ex);
}
```

openConnection()

BCA 6th

- **public URLConnection openConnection() throws IOException**

```
try {
      URL u = new URL("http://www.tufohss.edu.np");
 try {
 URLConnection uc = u.openConnection();
 InputStream in = uc.getInputStream();
 // read from the connection...
 } catch (IOException ex) {
 System.err.println(ex);
 }
} catch (MalformedURLException ex) {
 System.err.println(ex);
}
```

getContent()

BCA 6th

- **public final Object getContent() throws IOException**

```
try {
URL u = new URL("http://www.oreilly.com/graphics_new/animation.gif");
Object o = u.getContent();
System.out.println("I got a " + o.getClass().getName());
} catch (MalformedURLException ex) {
System.err.println(args[0] + " is not a parseable URL");
} catch (IOException ex) {
System.err.println(ex);
}
```

Retrieving Data From a URL

BCA 6th

- Now, create a new URL object and pass the desired URL that we want to access.
  *URL url = **new** URL(theUrl);*
- Now, using this url object, create a URLConnection object.
  URLConnection urlConnection = url.openConnection();
- Use the InputStreamReader and BufferedReader to read from the URL connection. BufferedReader bufferedReader = **new** BufferedReader();
  new InputStreamReader(urlConnection.getInputStream()));
- The readLine method of BufferedReader returns a String that we can access. If this string is null, it means we have reached the end of the document. **while** ((line = bufferedReader.readLine()) != **null**)
- Now, Append the string series that we have received as output from the URL to your StringBuilder object. content.append(line + "\n");
- print output: content.toString();

Example:

BCA 6th

```
URL url = new URL(theUrl); // creating a url object
URLConnection urlConnection = url.openConnection(); // creating a urlconnection object


// wrapping the urlconnection in a bufferedreader
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader( urlConnection.getInputStream()));
String line;
// reading from the urlconnection using the bufferedreader
while ((line = bufferedReader.readLine()) != null)
   {
     content.append(line + "\n");
   }
   bufferedReader.close();
System.out.println(content.toString());
```

Splitting a URL into Pieces

BCA 6th

URLs are composed of **FIVE pieces**:

- The scheme, also known as the protocol
- The authority
- The path
- The fragment identifier, also known as the section or ref
- The query string

E.g.

http://www.ibiblio.org:8080/javafaq/books/jnp/index.html? isbn=1565922069#toc

scheme = http,

authority = www.ibiblio.org:8080,

path = /javafaq/books/jnp/index.html, fragment identifier = toc,

query string = isbn=1565922069

Methods to access pieces

BCA 6th

- Read-only access to these parts of a URL is provided by nine public methods:
- getFile(), getHost(), getPort(), getProtocol(), getRef(), getQuery(), getPath(), getUserInfo(), and getAuthority().

Example: Splitting a URL into Pieces

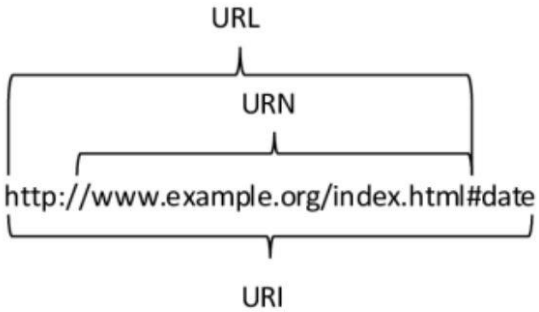
BCA 6th

```
URL u = new URL("http://www.ibiblio.org/wc/compositions.phtml?category=Piano");
System.out.println("The URL is " + u); // http://www.ibiblio.org/nywc/compositions.phtml?category=Piano
System.out.println("The scheme is " + u.getProtocol()); //http
System.out.println("The user info is " + u.getUserInfo()); //null
String host = u.getHost();
if (host != null) {
int atSign = host.indexOf('@');
if (atSign != -1) host = host.substring(atSign+1);
System.out.println("The host is " + host); //www.ibiblio.org
} else {
System.out.println("The host is null.");
}
System.out.println("The port is " + u.getPort()); //-1
System.out.println("The path is " + u.getPath()); //wc/compositions.phtml
System.out.println("The ref is " + u.getRef()); //null
System.out.println("The query string is " + u.getQuery()); //Piano
```

URI Class

BCA 6th

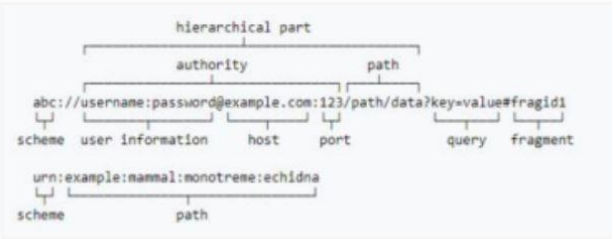
- A Uniform Resource Identifier (URI) is a sequence of characters used for identification of a particular resource.
- It enables for the interaction of the representation of the resource over the network using specific protocols.



URI Constructors

BCA 6th

- **URI(String str)** : Constructs a URI object by parsing the specified string.
- **URI(String scheme, String ssp, String fragment)** A component may be left undefined by passing null. Initially the result string is empty. If scheme is not null it is appended.
- **URI(String scheme, String userInfo, String host, int port, String path, String query, String fragment)**
- **URI(String scheme, String host, String path, String fragment)**
- **URI(String scheme, String authority, String path, String query, String fragment)**
  - scheme : string representing scheme
  - userInfo : userinfo of URI
  - host : host component of URI
  - port : listening port number
  - path : path of URI
  - query : String representing the query part
  - fragment :optional fragment



Constructing URI

BCA 6th

```
// Constructor to create a new URI by parsing the string
String uri = "http://www.ibiblio.org";
URI uriBase = new URI(uri);

// create() method
URI uriBase = URI.create(uri);
```

Resolving Relative URIs

BCA 6th

```
String uribase = "https://www.test.org/";
String urirelative = "languages/../java";
URI uriBase = new URI(uribase);

// create() method
URI uri = URI.create(str);
// toString() method
System.out.println("Base URI = " + uriBase.toString());//https://www.test.org/

URI uriRelative = new URI(urirelative);
System.out.println("Relative URI = " + uriRelative.toString());//languages/../java

  // resolve() method
URI uriResolved = uriBase.resolve(uriRelative);
System.out.println("Resolved URI = " + uriResolved.toString()) //https://www.geeksforgeeks.org/java
```

https://www.geeksforgeeks.org/java-net-uri-class-java/

URL Encoder

BCA 6th

- **Java.net.URLEncoder**
- This class is a utility class for HTML form encoding.
- Encoding makes the form of URL more reliable and secure.
- When the user request is triggered by a get method, the form parameters and their values are appended at the end of URL after a '?' sign.

Example

bca@network programming => bca%40network+programming

Rules when encoding a string

BCA 6th

- Alphanumeric characters and certain special characters such as '*', '_', '–' and '.' remains **unchanged**.
- **Spaces** are converted into '**+**' signs.
- All other characters are encoded by one or more bytes using the encoding scheme specified. They are converted in a three character string of the form **%xy**, where xy represents the hexadecimal representation of the encoding character.

Example

bca@ambition academy => bca%40ambition+academy

Method: encode()

BCA 6th

(a) Syntax : public static String encode(String s)

Parameters :

s : String to be encoded

```
URLEncoder.encode("hello world", "UTF-8")
```

(b) Syntax : public static String encode(String s, String enc)

Parameters :

s : string to be encoded

enc : encoding to be used e.g. UTF-8

```
URLEncoder.encode("hello world", "UTF-8")
```

Example Code

BCA 6th

```
// base URL
String baseurl = "https://www.ambition.edu.np/?q=";

// String to be encoded
String query = "info@gmail for bca";

System.out.println("URL without encoding :");
URL url = new URL(baseurl + query);
System.out.println(url);

// encode() method
System.out.println("URL after encoding :");
url = new URL(baseurl + URLEncoder.encode(query, "UTF-8"));
System.out.println(url);
```


URL Decoder

BCA 6th

- **Java.net.URLDecoder class**
- utility class for HTML form decoding.
- It just performs the reverse of what URLEncoder class do, i.e. given an encoded string, it decodes it using the scheme specified.

Example

bca%40network+programming => bca@network programming

Steps while decoding the strings

BCA 6th

- Alphanumeric characters and certain special characters such as '*', '_', '–' and '.' remains unchanged.
- '+' signs are converted into spaces.
- All other characters are decoded using the encoding scheme specified. The string of the form **%xy**, is converted to the character whose encoding would have resulted in this three character representation.

Example

bca%40ambition+academy => bca@ambition academy

decode()

BCA 6th

(a) Syntax :public static String decode(String s)

Parameters :

s : encoded string to be decoded

```
URLDecoder.decode("hello+world")
```

(b) Syntax :public static String decode(String s, String enc)

Parameters :

s : string to be decoded

enc : encoding to be used e.g. UTF-8

```
URLDecoder.decode("hello+world", "UTF-8")
```

Example Code

BCA 6th

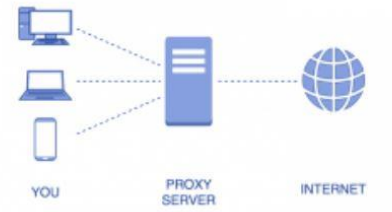
```
// encoded string
String encodedString = “info%40gmail+for+bca";
System.out.println("Encoded String :");
System.out.println(encodedString);

// decode() method
System.out.println("Decoded String :");
System.out.println(URLDecoder.decode(encodedString, "UTF-8"));
```

Proxies

BCA 6th

- Proxy means **‘in place of’**, representing’ or ‘**in place of**’ or ‘**on behalf of**’
- A real world example can be a cheque or credit card is a proxy for what is in our bank account.
- Proxy pattern does – “**Controls and manage access to the object they are protecting**“.



System Properties

BCA 6th

- For basic operations, all you have to do is set a few system properties to point to the addresses of your local proxy servers.
- If you are using a **HTTP proxy**, set http.proxyHost to the domain name or the IP address of your proxy server and http.proxyPort to the port of the proxy server (the default is 80).
- System.setProperty("http.proxyHost", "192.168.254.254");
- System.setProperty("http.proxyPort", "9000");
- System.setProperty("http.nonProxyHosts","java.oreilly.com|xml.oreilly.com");

Proxy Class

BCA 6th

- The Proxy class allows more fine-grained control of proxy servers from within a Java program.
- Specifically, it allows you to choose different proxy servers for different remote hosts.
- The proxies themselves are represented by instances of the java.net.Proxy class.
- **Example:**

```
SocketAddress address = new InetSocketAddress("proxy.example.com", 80);
Proxy proxy = new Proxy(Proxy.Type.HTTP, address);
```

Proxy Selector

BCA 6th

- **Each running virtual machine** has a single java.net.ProxySelector object it uses to *locate the proxy server for different connections*
- To change the Proxy Selector, pass the new selector to the static **ProxySelector.setDefault()** method, like so:

```
ProxySelector selector = new LocalProxySelector(); // returns list of proxies

ProxySelector.setDefault(selector);
```

Communicating with Server-Side Programs Through GET

BCA 6th

- The URL class makes it easy for Java applets and applications to communicate with serverside programs such as CGIs, servlets, PHP pages, and others that use the GET method.

```
<form name="search" action="http://www.google.com/search" method="get">
        <input name="q" />
        <input type="submit" value="Search" />
</form>
```

Basic Syntax

BCA 6th

http://www.google.com/search?q=computer

```
QueryString query = new QueryString();

 query.add("q", target);

 URL u = new URL("http://www.google.com/search?" + query);

... write code for reading webpage
```

Accessing Password-Protected Site

BCA 6th

- Java's URL class **can access** sites that use **HTTP authentication**, though you'll of course need to *tell it what username and password to use.*
- **cookie-based authentication** is more challenging, not least because this varies a lot from one site to another

Authenticator Class: Authenticator()

BCA 6th

- the java.net package includes an Authenticator class you can use to provide a username and password for sites that protect themselves using HTTP authentication:

  public abstract class Authenticator extends Object

- **Methods**

```
Authenticator.setDefault(new DialogAuthenticator());
// Sets the authenticator to be used when a HTTP server requires authentication.
```

Methods from the Authenticator superclass

BCA 6th

- protected final InetAddress getRequestingSite() // requesting for the authorization,
- protected final int getRequestingPort()
- protected final String getRequestingProtocol()
- protected final String getRequestingPrompt()
- protected final String getRequestingScheme()
- protected final String getRequestingHost()
- protected final String getRequestingURL()
- protected RequestorType getRequestorType() // requester is a Proxy or a Server.

Example: Methods from the Authenticator

BCA 6th

ClassAuthenticator obj1 = **new** ClassAuthenticator();

Authenticator.setDefault(**new** ClassAuthenticator());

obj1.getPasswordAuthentication() ;

```
public static class ClassAuthenticator extends Authenticator
 {
    protected PasswordAuthentication getPasswordAuthentication()
    {
          System.out.println("Port Requesting :" + getRequestingPort());
          String username = "javaTpoint";
          String password = "java";
          return new PasswordAuthentication(username, password.toCharArray());
    }
```

PasswordAuthentication Class

BCA 6th

- **PasswordAuthentication** is a very simple final class that supports **two read-only properties**: username and password.
- **Syntax**

public PasswordAuthentication(String userName, char[] password)

- Each is **accessed** via a getter method:

public String getUserName( )

public char[] getPassword( )

Example: PasswordAuthentication Class

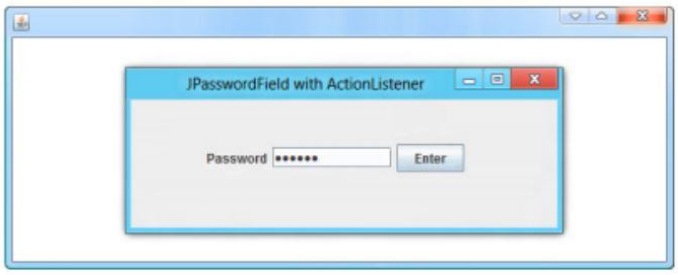
BCA 6th

```
String userName = “user";
char[] password = { ‘p', ‘a', ‘s', ‘s' };
PasswordAuthentication auth = new PasswordAuthentication(userName, password);
System.out.println("UserName: " + auth.getUserName());
System.out.println( "Password: "   + passwordAuthentication.getPassword());
```

JPasswordField CLASS

BCA 6th

- One useful tool for asking users for their passwords in a more or less secure fashion is the JPasswordField component from Swing:
- public class JPasswordField extends JTextField



Example: JPasswordField

BCA 6th

```
import javax.swing.*;
public class PasswordFieldExample {
   public static void main(String[] args) {
   JFrame f=new JFrame("Password Field Example");
    JPasswordField value = new JPasswordField();
    JLabel l1=new JLabel("Password:");
      l1.setBounds(20,100, 80,30);
       value.setBounds(100,100,100,30);
         f.add(value);  f.add(l1);
         f.setSize(300,300);
         f.setLayout(null);
         f.setVisible(true);
}
}
```

BCA 6th

The End