Network Programming

[CACS355]

BCA 6th Sem

Er. Sital Prasad Mandal

(Email : info.sitalmandal@gmail.com)

Bhadrapur, Jhapa, Nepal

https://networkprogam-mmc.blogspot.com/

https://networkprogam-mmc.blogspot.com/

Unit-3

URLs and URIs.

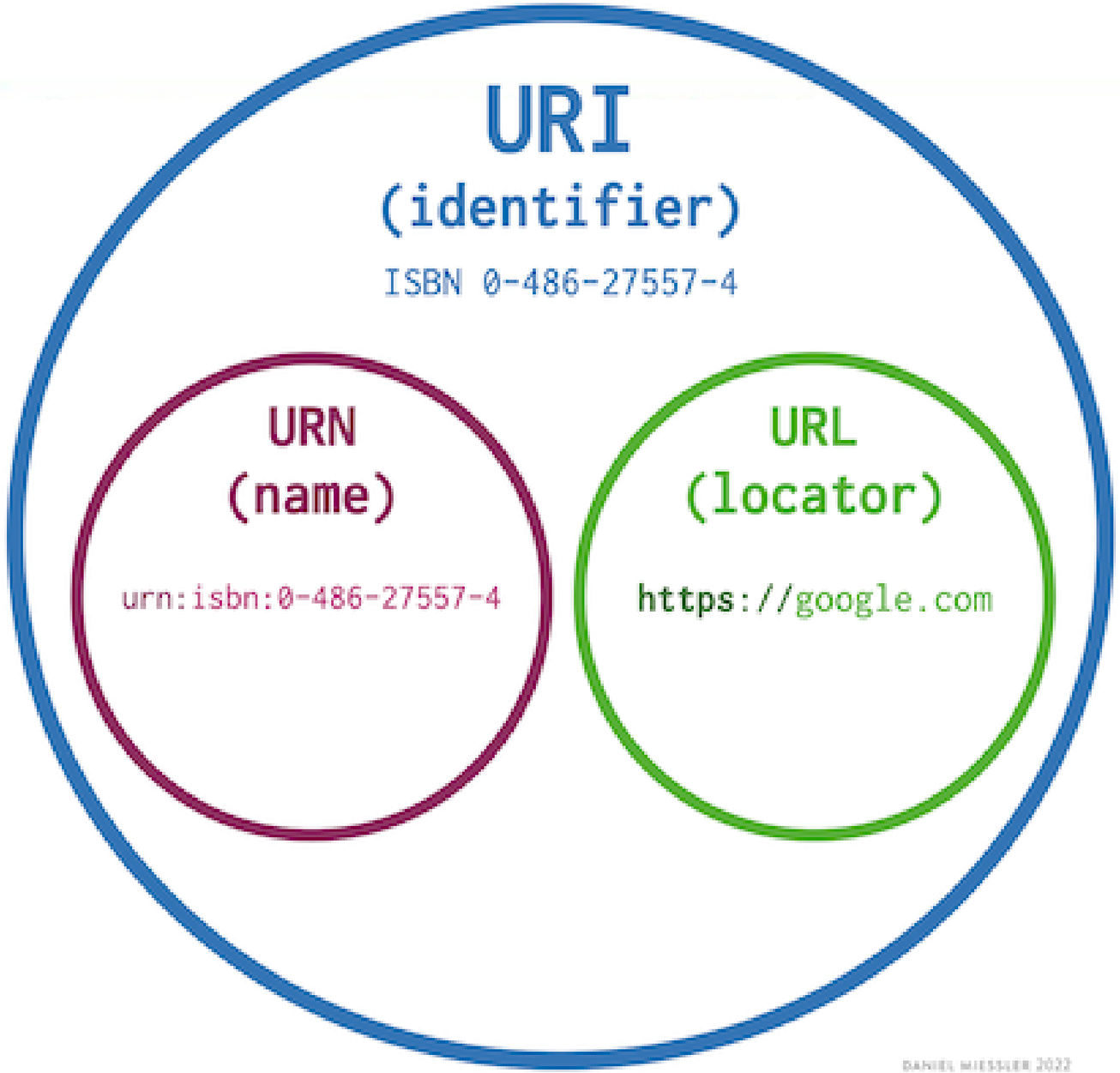
3.1	URIs
	URLs
	Relative URLs
3.2	The URL Class
	Creating New URLs
	Retrieving Data from a URL
	Splitting a URL into Pieces
	Equality and Comparison
	Conversion
3.4	X-www-form-urlencoded: URL Encoder & URL Decoder
3.5	Proxies

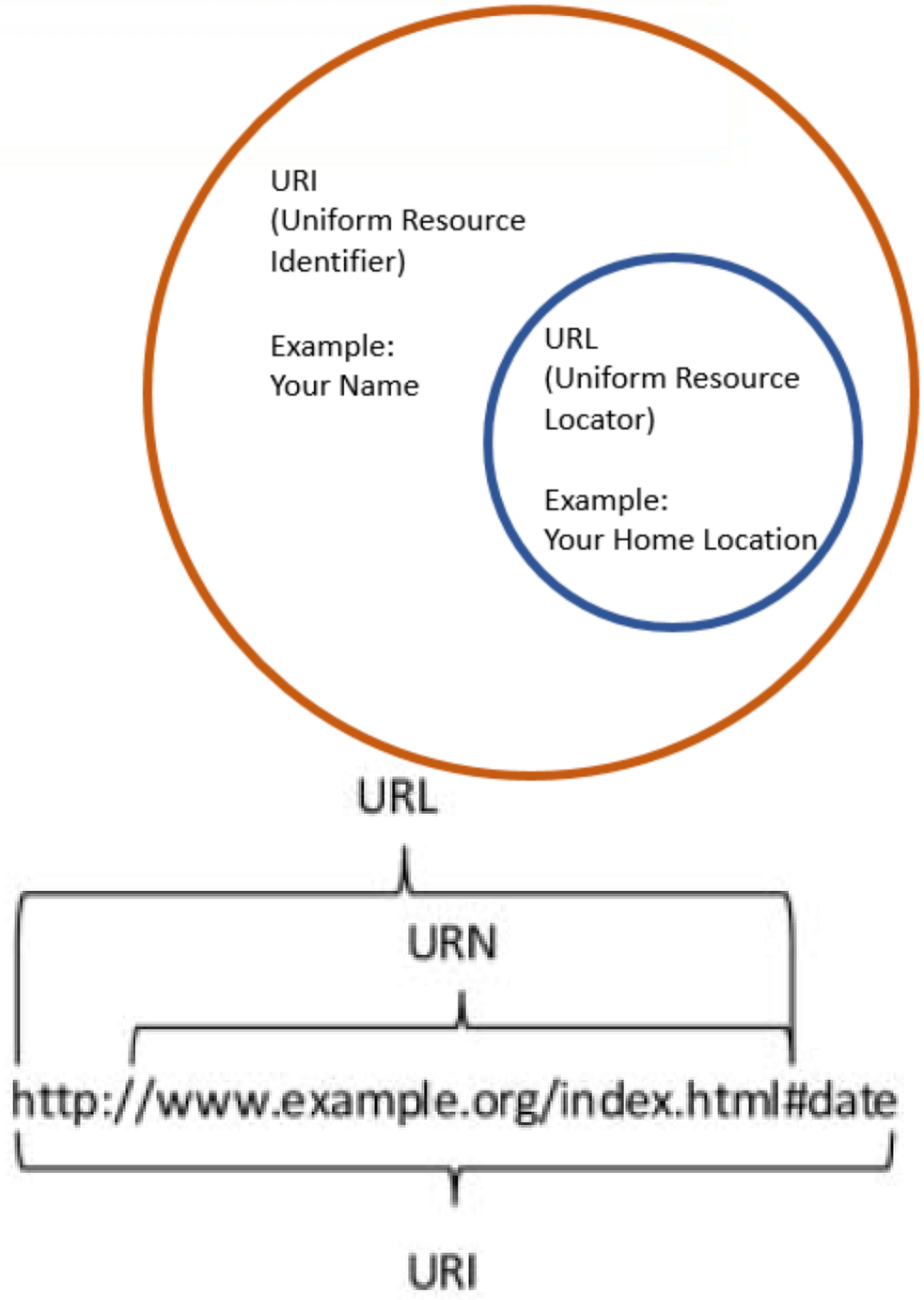
3.3	The URI Class
	Constructing a URI
	The Parts of the URI
	Resolving Relative URIs
	Equality and Comparison
	String Representations
3.6	Communicating with Server-Side Programs Through GET
3.7	Access Password-Protected Sites
	Authenticator Class
	Password Authentication Class
	JPasswordField Class

https://networkprogam-mmc.blogspot.com/

Unit-3

URLs and URIs





Unit-3

URLs and URIs

URLs, URIs, and URNs

- URL is a Uniform Resource Locator, tells you the how and where of something
  - http://www.wrox.com/remtitle.cgi?isgn=0470114878
- URN is a Uniform Resource Name, is simply a unique name
  - urn:[namespace identifier]:[namespace specific string]
  - urn:isbn:9780470114872
- URI is a Uniform Resource Identifier, is URL or URN

https://networkprogam-mmc.blogspot.com/

Unit-3

URL

The syntax of a URL is:

protocol://userInfo@host:port/path?query#fragment

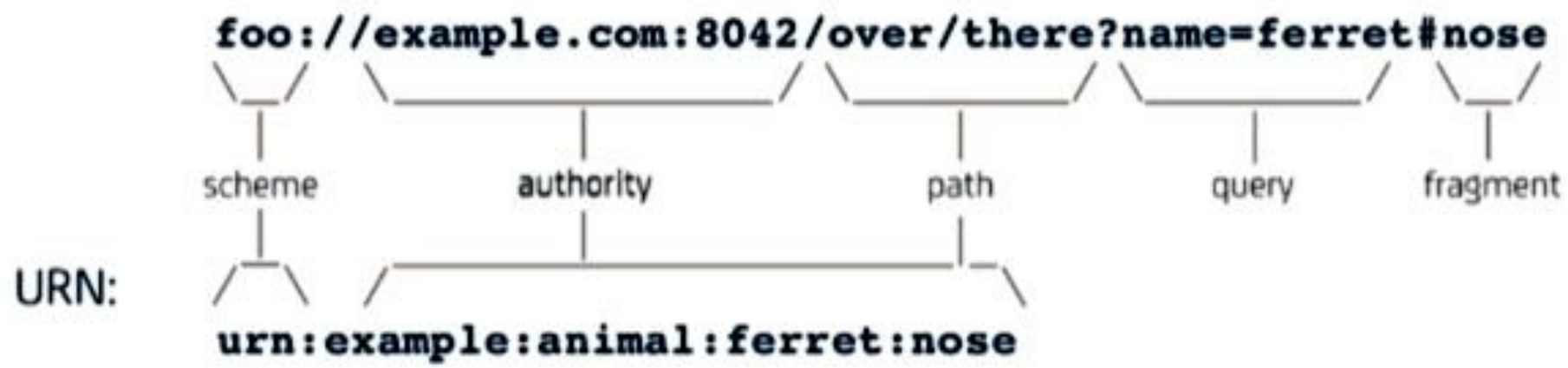
http://www.cafeaulait.org/javafaq.html#xtocid1902914

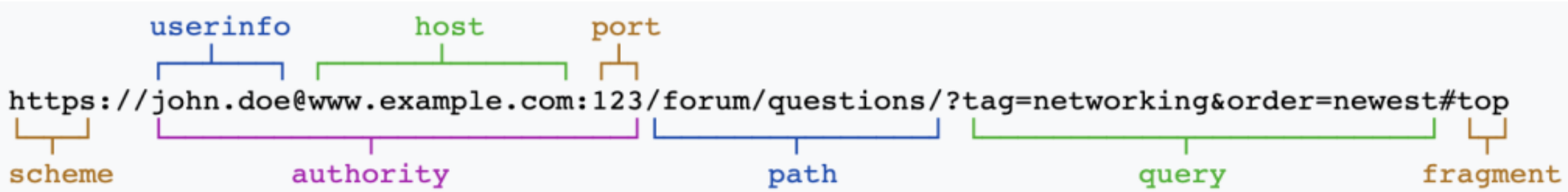
https://networkprogam-mmc.blogspot.com/

Unit-3

URIs

The structure of URIs





https://networkprogam-mmc.blogspot.com/

Unit-3

URLs and URIs

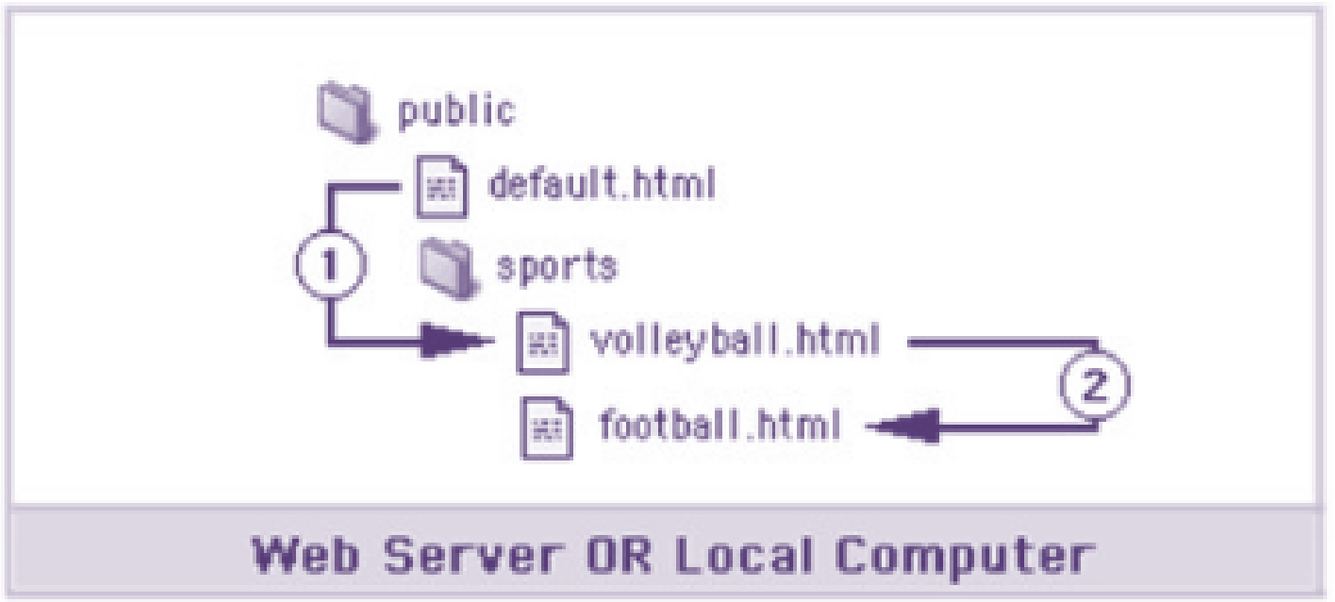
URL	URI
URL is used to describe the identity of an item.	URI provides a technique for defining the identity of an item.
URL links a web page, a component of a web page or a program on a web page with the help of accessing methods like protocols.	URI is used to differentiate one resource from other regardless of the method used.
URL provides the details about what type of protocol is to be used.	URI doesn't contains the protocol specification.
URL is a type of URI.	URI is the superset of URL.
mailto://bob@fb.com	bob@google.com
like http://, ftp://, or mailto://	urn:isbn:0-486-27557-4

Unit-3

Relative URLs

- When you are linking your Web site together, use *relative URLs*.
- A relative URL gives the path to the file to which you wish to link, *relative* to the page in which the link appears.

① href="sports/volleyball.html"
② href="football.html"




```
<a href="http://www.uweb.edu/~jones/sports/football.html">
    Check out my football page
</a>
```

Unit-3

The URL Class

3.2 The URL Class

1. Creating New URLs
2. Retrieving Data from a URL
3. Splitting a URL into Pieces
4. Equality and Comparison
5. Conversion

Unit-3

The URL Class

Creating New URLs

Instances of java.net.URL. The constructors differ in the information they require:

1. public URL(String url) throws MalformedURLException
2. public URL(String protocol, String hostname, String file) throws MalformedURLException
3. public URL(String protocol, String host, int port, String file) throws MalformedURLException
4. public URL(URL base, String relative) throws MalformedURLException

Unit-3

The URL Class

(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/");

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

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

(d) 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");

Unit-3

The URL Class

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);
}
```

Unit-3

The URL Class

Retrieving Data from a URL

1. public InputStream openStream() throws IOException
2. public URLConnection openConnection() throws IOException
3. public URLConnection openConnection(Proxy proxy) throws IOException
4. public Object getContent() throws IOException
5. public Object getContent(Class[] classes) throws IOException

Unit-3

The URL Class

Retrieving Data from a URL

- 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.

Unit-3

The URL Class

Retrieving Data from a URL

1. public InputStream openStream() throws IOException
2. public URLConnection openConnection() throws IOException
3. public URLConnection openConnection(Proxy proxy) throws IOException
4. public Object getContent() throws IOException
5. public Object getContent(Class[] classes) throws IOException

Unit-3

The URL Class

Retrieving Data from a URL

public final InputStream openStream() throws IOException

Opens a connection to this URL and returns an InputStream for reading from that connection.

```
try {
    String location = "https://lolcats.com/";
    URL url = new URL(location);
    InputStream is = url.openStream();
    int c;
    char data;
    while ((c = is.read()) != -1) {
        data = (char) c;
        System.out.print(data);
    }
    is.close();
} catch (IOException ex) {
    System.out.println(ex);
}
```

```
BufferedReader br = new BufferedReader(new InputStreamReader(is));
String line;
while((line=br.readLine()) != null){
System.out.println(line);
}
```

```
BufferedReader br = new BufferedReader(new InputStreamReader(is));
{
    br.lines().forEach(System.out::println);
}
```

Unit-3

The URL Class

Retrieving Data from a URL

2. public URLConnection openConnection() throws IOException

Returns a URLConnection instance that represents a connection to the remote object referred to by the URL.

```
try {
    String location = "https://lolcats.com/";
    URL url = new URL(location);
    URLConnection conn = url.openConnection();
    InputStream is = conn.getInputStream();
    BufferedReader br = new BufferedReader(new
InputStreamReader(is));
    {
        br.lines().forEach(System.out::println);
    }
} catch (IOException ex) {
    System.out.println(ex);
}
```

Unit-3

The URL Class

Retrieving Data from a URL

3. public final Object getContent() throws IOException

getContent() returns the contents of the URL.
Depends on the type of data returned by the getContent() method, it refers to specific kind of object.

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

        String location = "http://www.spm.com.np";
        String location1 = “Bargument.PNG";
        URL url = new URL(location);

        Object content = url.getContent();
        System.out.println(content.getClass().getName());
    }
}
```

Output 1
sun.net.www.protoco.http.HttpURLConnection$HttpInputStream
Output 2
For the above image, you will get following output.
sun.awt.image.URLImageSource

Unit-3

The URL Class

Retrieving Data from a URL

4. public final Object getContent(Class[] classes) throws IOException

It is overloaded version of getContent method. By using this method, you can choose which class you'd like the content to be returned as. This method returns the URL content in the first available format, that user specified.

```
        try {
            String location = "https://spm.com.np";
            URL url = new URL(location);

            Class<?>[] types = new Class[3];
            types[0] = String.class;
            types[1] = Reader.class;
            types[2] = InputStream.class;
            Object o = url.getContent(types);

        System.out.println(o.getClass().getName());
        } catch (IOException e){
            System.out.println(e);
        }
```

Output

sun.net.www.protocol.http.HttpURLConnection$HttpInputStream

Unit-3

The URL Class

Retrieving Data from a URL

4. public final Object getContent(Class[] classes) throws IOException

It is overloaded version of getContent method. By using this method, you can choose which class you'd like the content to be returned as. This method returns the URL content in the first available format, that user specified.

```
        try {
            String location = "https://spm.com.np";
            URL url = new URL(location);

            Class<?>[] types = new Class[3];
            types[0] = String.class;
            types[1] = Reader.class;
            types[2] = InputStream.class;
            Object o = url.getContent(types);

        System.out.println(o.getClass().getName());
        } catch (IOException e){
            System.out.println(e);
        }
```

Output

sun.net.www.protocol.http.HttpURLConnection$HttpInputStream

Unit-3

The URL Class

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

Unit-3

The URL Class

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

        String location =
"https://docs.oracle.com/javase/7/docs/api/java/net/URL.html#getCont
ent()";
        URL url = new URL(location);

        System.out.println("Authority : " +url.getAuthority());
        System.out.println("Deafult Port : " +url.getDefaultPort());
        System.out.println("File : " +url.getFile());
        System.out.println("Host : " +url.getHost());
        System.out.println("Path : " +url.getPath());
        System.out.println("Port : " +url.getPort());
        System.out.println("Protocol : " +url.getProtocol());
        System.out.println("Query : " +url.getQuery());
        System.out.println("Reference (Anchor) : " +url.getRef());
        System.out.println("User Info : " +url.getUserInfo());
    }
}
```

Unit-3

The URL Class

Equality and Comparison

Example 5-5. Are http://www.ibiblio.org and http://ibiblio.org the same?

```
import java.net.*;
public class URLEquality {
public static void main (String[] args) {
try {
URL www = new URL ("http://www.ibiblio.org/");
URL ibiblio = new URL("http://ibiblio.org/");
if (ibiblio.equals(www)) {
System.out.println(ibiblio + " is the same as " + www);
} else {
System.out.println(ibiblio + " is not the same as " + www);
}
} catch (MalformedURLException ex) {
System.err.println(ex);
}
}
}
```

O/P:

http://ibiblio.org/ is the same as http://www.ibiblio.org/

Unit-3

The URL Class

Equality and Comparison

The sameFile() does not consider the fragment identifier.

Here's a fragment of code that uses `sameFile()` to compare two URLs:

```
URL u1 = new URL("http://www.ncsa.uiuc.edu/HTMLPrimer.html#GS");
URL u2 = new URL("http://www.ncsa.uiuc.edu/HTMLPrimer.html#HD");
if (u1.sameFile(u2)) {
  System.out.println(u1 + " is the same file as \n" + u2);
} else {
  System.out.println(u1 + " is not the same file as \n" + u2);
}
```

The output is:

```
http://www.ncsa.uiuc.edu/HTMLPrimer.html#GS is the same file as
http://www.ncsa.uiuc.edu/HTMLPrimer.html#HD
```

https://networkprogam-mmc.blogspot.com/

Unit-3

The URI Class

3.3	The URI Class
	Constructing a URI
	The Parts of the URI
	Resolving Relative URIs
	Equality and Comparison
	String Representations

Unit-3

The URI Class

URI stands for Uniform Resource Identifier. A uniform resource identifier (URI) is a string of characters used to identify a name of a resource.

URI's are defined as two types URL and URN.

Uniform Resource Locator (URL): This is an address used to identify network/resource locations.
Example: http://gmail.com

Uniform Resource Name(URN): This is persistent name, which is address independent. A URN can be used to identify a resource without implying its location or how to access it.
Example: URN:ISBN:0-395-36341-1

Above one specifies the unique reference within the International Standard Book Number (ISBN) identifier system. It references a resource, but doesn't specify how to obtain an actual copy of the book.

Unit-3

The URI Class

URI stands for Uniform Resource Identifier. A uniform resource identifier (URI) is a string of characters used to identify a name of a resource.

URI's are defined as two types URL and URN.

Uniform Resource Locator (URL): This is an address used to identify network/resource locations.
Example: http://gmail.com

Uniform Resource Name(URN): This is persistent name, which is address independent. A URN can be used to identify a resource without implying its location or how to access it.
Example: URN:ISBN:0-395-36341-1

Above one specifies the unique reference within the International Standard Book Number (ISBN) identifier system.

General syntax for URI

[scheme:]scheme-specific-part[#fragment]

https://networkprogam-mmc.blogspot.com/

Unit-3

The URI Class

Constructors URI object:

1. **public URI(String uri) throws URISyntaxException**
2. **public URI(String scheme, String schemeSpecificPart, String fragment) throws URISyntaxException**
3. **public URI(String scheme, String host, String path, String fragment) throws URISyntaxException**
4. **public URI(String scheme, String authority, String path, String query,** String fragment) **throws URISyntaxException**
5. **public URI(String scheme, String userInfo, String host, int port,** String path, String query, String fragment) **throws URISyntaxException**

Unit-3

The URI Class

URI(String str)

Construts URI, by parsing given string. If the string passed to this constructor, is not followed URI standrads, then it throws URISyntaxException.

```
public class Main {
  public static void main(String args[]) throws URISyntaxException {
    URI uri1 = new URI("http://www.google.com");
    URI uri2 = new URI("URN:ISBN:0-395-36341-1");

    System.out.println(uri1);
    System.out.println("Authority : " + uri1.getAuthority());
    System.out.println("Fragment : " + uri1.getFragment());
    System.out.println("Host : " + uri1.getHost());
    System.out.println("Scheme : " + uri1.getScheme());
    System.out.println("*********************************");

    System.out.println(uri2);
    System.out.println("Authority : " + uri2.getAuthority());
    System.out.println("Fragment : " + uri2.getFragment());
    System.out.println("Host : " + uri2.getHost());
    System.out.println("Scheme : " + uri2.getScheme());
  }
}
```

Output

```
http://www.google.com
Authority : www.google.com
Fragment : null
Host : www.google.com
Scheme : http
*********************************
URN:ISBN:0-395-36341-1
Authority : null
Fragment : null
Host : null
Scheme : URN
null
```

Unit-3

The URI Class

URI(String scheme, String ssp, String fragment)
Constructs URI from the components scheme (http, ftp, smtp etc.,), ssp (Scheme Specific part), fragment. Final result like *"scheme:ssp#fragment".*

```
public class Main {
  public static void main(String args[]) throws URISyntaxException {
    URI uri1 = new URI("http", "//www.google.com", "search1");

    System.out.println(uri1);
    System.out.println("Authority : " + uri1.getAuthority());
    System.out.println("Fragment : " + uri1.getFragment());
    System.out.println("Host : " + uri1.getHost());
    System.out.println("Scheme : " + uri1.getScheme());
  }
}
```

```
http://www.google.com#search1
Authority : www.google.com
Fragment : search1
Host : www.google.com
Scheme : http
```

Unit-3

The URI Class

public URI(String scheme, String host, String path, String fragment) throws URISyntaxException

Constructs URI from the components scheme (http, ftp, smtp etc.,), host (host name), path, fragment. Final result like *"scheme:host/path#fragment"*.

```
public class Main {
  public static void main(String args[]) throws URISyntaxException {
    URI uri = new URI("http", "www.docs.oracle.com", "/javase/7/docs/api/java/net/URI.html", "URI");

    System.out.println(uri);
    System.out.println("Authority : " + uri.getAuthority());
    System.out.println("Fragment : " + uri.getFragment());
    System.out.println("Host : " + uri.getHost());
    System.out.println("Scheme : " + uri.getScheme());
  }
}
```

Output

http://www.docs.oracle.com/javase/7/docs/api/java/net/URI.html#

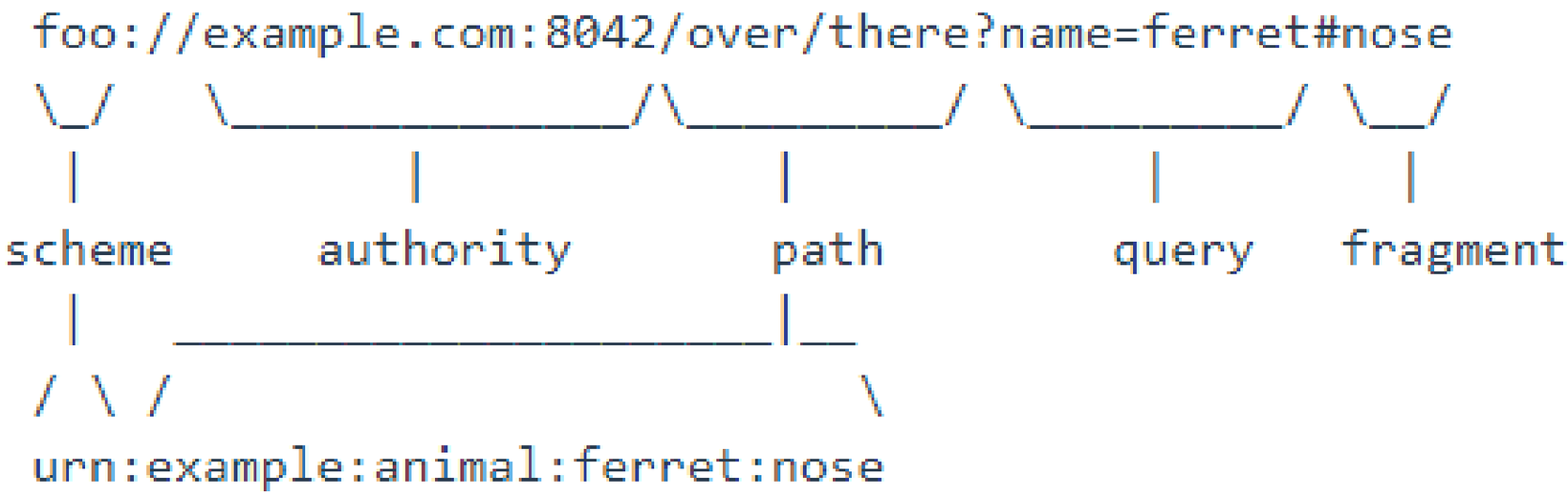
https://networkprogam-mmc.blogspot.com/

Unit-3

The URI Class

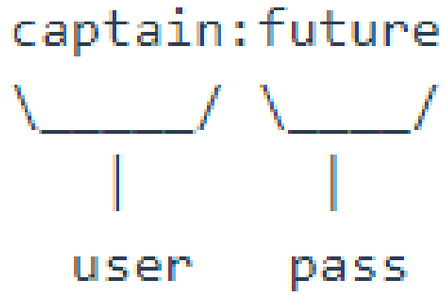
The Parts of the URI

URI string is composed of 8 components and 5 parts:



URI authority part composed of up to 3 parts:

userinfo part is composed of the `user` and the `pass`



https://networkprogam-mmc.blogspot.com/

Unit-3

The URI Class

Example 5-6. The parts of a URI

```
import java.net.*;
public class URISplitter {
public static void main(String args[]) {
try {
URI u = new URI(args[i]);
System.out.println("The URI is " + u);
if (u.isOpaque()) {
System.out.println("This is an opaque URI.");
System.out.println("The scheme is " + u.getScheme());
System.out.println("The scheme specific part is "
+ u.getSchemeSpecificPart());
System.out.println("The fragment ID is " + u.getFragment());
} else {
System.out.println("This is a hierarchical URI.");
System.out.println("The scheme is " + u.getScheme());
try {
u = u.parseServerAuthority();
System.out.println("The host is " + u.getHost());
System.out.println("The user info is " + u.getUserInfo());
System.out.println("The port is " + u.getPort());
} catch (URISyntaxException ex) {
// Must be a registry based authority
System.out.println("The authority is " + u.getAuthority());
}
System.out.println("The path is " + u.getPath());
System.out.println("The query string is " + u.getQuery());
System.out.println("The fragment ID is " + u.getFragment());
}
} catch (URISyntaxException ex) {
System.err.println(args[i] + " does not seem to be a URI.");
}
System.out.println();
}
}
```

```
% java URISplitter tel:+1-800-9988-9938
http://www.xml.com/pub/a/2003/09/17/stax.html#id=_hbc
urn:isbn:1-565-92870-9
The URI is tel:+1-800-9988-9938
This is an opaque URI.
The scheme is tel
The scheme specific part is +1-800-9988-9938
The fragment ID is null
```

isOpaque: A Boolean value indicating whether the title is empty

Unit-3

The URI Class

Resolving Relative URIs

```
String uribase ="http://www.example.com/";
String urirelative = "images/logo.png";
URI uriBase = new URI(uribase);

// create() method
URI uri = URI.create(str);
// toString() method
System.out.println("Base URI = " + uriBase.toString());   "http://www.example.com/";

URI uriRelative = new URI(urirelative);
System.out.println("Relative URI = " + uriRelative.toString());  "images/logo.png";

  // resolve() method
URI uriResolved = uriBase.resolve(uriRelative);
System.out.println("Resolved URI = " + uriResolved.toString())
```

http://www.example.com/images/logo.png.

Unit-3

X-www-form-urlencoded: URL Encoder

URLEncoder class is used for HTML form encoding. This class contains static methods for converting a String to the application/x-www-form-urlencoded MIME format. URLEnocder class converts any non alpha-numeric characters (except ".", "-", "*", and "_") to %sequences.

```
public class URLEncoder {
public static void main(String args[]) throws URISyntaxException, UnsupportedEncodingException {
                String data = "Simple text data!@#$";
                String encodeData = URLEncoder.encode(data, "UTF-8");
                System.out.println(encodeData);
        }
}
```

Output

```
Simple+text+data%21%40%23%24
```

Why should we encode URLs?

Consider for example, in a query string, the ampersand (&) is used as a separator between key and value pairs (name=krishna&age=26). If any of the parameter value has an ampersand in it, it would look like the separator between the end of a value and the beginning of the next key. So for special characters like this, we use URL encoding so that we can be sure that the data is unambiguously encoded.

Unit-3

X-www-form-urlencoded: URL Decoder

URLDecoder class is used to decodes strings encoded in x-www-form-url-encoded format.

```
import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;

public class URLDecoder{
public static void main(String args[]) throws UnsupportedEncodingException {
      String data = "http://www.mysite.com/?video=funny%20cat%20plays%20piano.";
      String result = URLDecoder.decode(data, "UTF-8");
      System.out.println(result);
       }
}
```

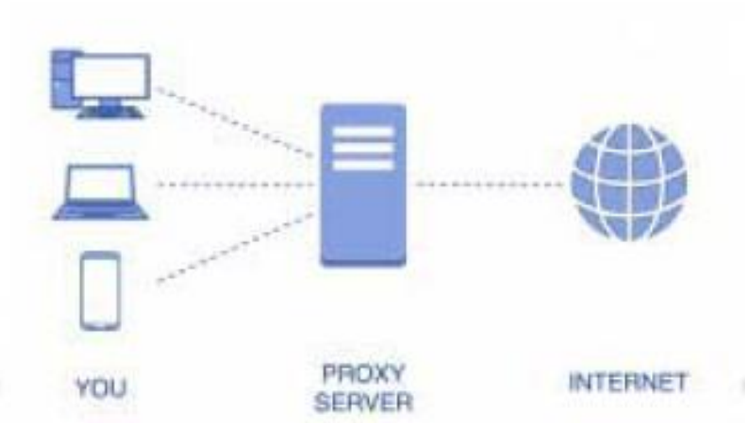
Output

```
http://www.mysite.com/?video=funny cat plays piano.
```

Unit-3

Proxies

- 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**".



Unit-3

Proxies

System Properties

- 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");

Unit-3

Proxies

Proxy Class

- 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);
```

Unit-3

Proxies

Proxy Selector

- **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);
```

Unit-3

Communicating with Server-Side Programs Through GET

- 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>
```

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

Unit-3

Access Password-Protected Sites

- 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

Unit-3

Access Password-Protected Sites

Authenticator Class: Authenticator()

- 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.
```

Unit-3

Access Password-Protected Sites

Methods from the Authenticator superclass

- 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.

Unit-3

Access Password-Protected Sites

PasswordAuthentication Class

- **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( )
```

Unit-3

Access Password-Protected Sites

JPasswordField CLASS

- 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