Programmation Réseau

URL

Jean-Baptiste.Yunes@liafa.jussieu.fr

UFR Informatique

2011-2012

URI - URL - URN

- Défini dans la RFC 3986 « Uniform Resource Identifier (URI): Generic syntax », 01/2005

  - nouvelle version de la RFC 2396 de 08/1998

  - extension de la RFC 1738 « Uniform Resource Locators (URL) » de 12/1994

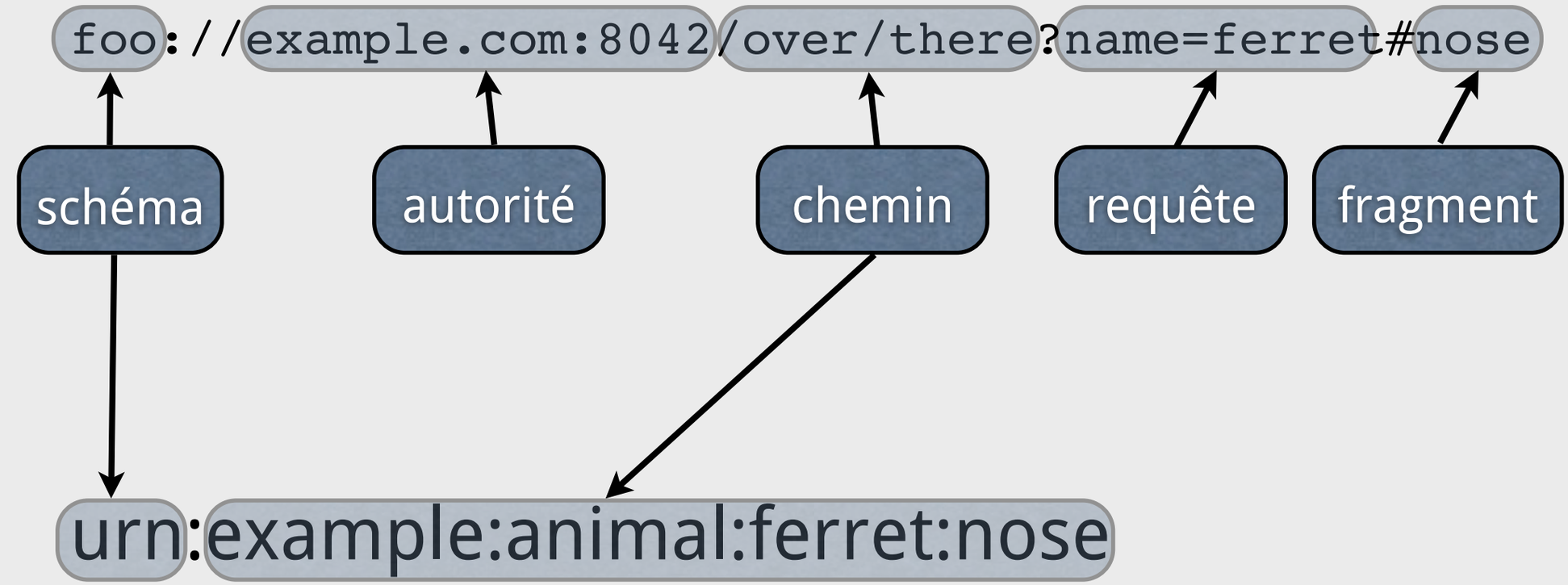
- le système de désignation de documents via le Web

- URI : Uniform Resource Identifier
- URL : Uniform Resource Locator
- URN : Uniform Resource Name
- Uniform : car ces nommages permettent de s'affranchir des mécanismes d'accès (protocoles)
- Resource : ce que l'on souhaite, rien n'est imposé sur ce qu'est une resource
- Identifier : permet la distinction entre resources différentes

université PARIS DIDEROT PARIS 7

- Examples (extraits de la RFC) :
  - ftp://ftp.is.co.za/rfc/rfc1808.txt
  - http://www.ietf.org/rfc/rfc2396.txt
  - ldap://[2001:db8::7]/c=GB?objectClass?one
  - mailto:John.Doe@example.com
  - news:comp.infosystems.www.servers.unix
  - tel:+1-816-555-1212
  - telnet://192.0.2.16:80/
  - urn:oasis:names:specification:docbook:dtd:xml:4.1.2

- Les URIs se décomposent :



java.net.URI

- outre les divers constructeurs les méthodes suivantes permettent d'extraire les composants intéressants de l'URI

  - `String getScheme()`
  - `String getAuthority()`
  - `String getPath()`
  - `String getQuery()`
  - `String getFragment()`

- plus diverses autres (raffinement)…

- cette classe est assez peu employée en Java

java.net.URL

- permet en particulier d'établir une liaison avec la ressource en question, via :
  - `URLConnection openConnection();`
    - qui permet d'envisager la manipulation de la connexion sous-jacente
  - `InputStream openStream();`
    - qui permet d'obtenir la ressource sous-jacente

```
public class TestURL {
  public static void main(String []args) {
    try {
      // construit une URL
      URL url = new URL(args[0]);
      // l'ouvre comme un flot de lecture
      BufferedReader bf = new BufferedReader(
        new InputStreamReader(url.openStream()));
      // on lit tout ce qu'il est possible de lire
      String s = bf.readLine();
      while (s!=null) {
        System.out.println(s); s = bf.readLine();
      }
      // on ferme
      bf.close();
    } catch(Exception e) {
      e.printStackTrace(); System.exit(1);
    }
  }
}
```

`java.net.URLConnection`

- permet de récupérer les **en-têtes** séparement

- du **contenu**

- vers lequel on peut possiblement lire et écrire via :

  - des flots Java

- pour la lecture, on peut extraire l'`Object` correspondant à la ressource

- les en-têtes peuvent récupérés via :

```
Map<String,List<String>> getHeaderFields();
```

  - attention : certains en-têtes n'ont pas de label, une clé `null` leur correspond

```
public class TestURL2 {
  private static void printHeaders(Map<String,List<String>> headers) {
    // liste des clés d'en-têtes
    Iterator<String> keys = headers.keySet().iterator();
    while (keys.hasNext()) {
      String fieldName = keys.next();
      if (fieldName!=null) System.out.print(fieldName+": ");
      // liste des valeurs associées
      Iterator<String> it = headers.get(fieldName).iterator();
      while (it.hasNext()) {
        System.out.println(it.next());
      }
    }
  }
  public static void main(String []args) {
    try {
      URL url = new URL(args[0]);
      URLConnection urlc = url.openConnection();
      printHeaders(urlc.getHeaderFields());
    } catch(Exception e) {
      e.printStackTrace(); System.exit(1);
    }
  }
}
```

- des informations peuvent être récupérées :
  - qui concernent la connexion elle-même
    - propriétés
  - qui concernent la ressource
    - méta-information

- propriétés de la connexion :
  - `boolean getAllowUserInteraction();`
  - `int getConnectTimeout();`
  - `boolean getDoInput();`
  - `boolean getDoOutput();`
  - `int getReadTimeout();`
  - `boolean getUseCaches();`

- méta-information :
  - `String getContentEncoding();`
  - `int getContentLength();`
  - `String getContentType();`
  - `long getDate();`
  - `long getExpiration();`
  - `long getIfModifiedSince();`
  - `long getLastModified();`

- pour le contenu :
  - soit au plus bas-niveau avec :
    - `InputStream getInputStream();`
    - `OutputStream getOutputStream();`
  - soit en instanciant une classe appropriée si possible :
    - `Object getContent();`

```
public class TestURL3 {
  private static String[] hexArray = { "00", …, "FF" };
  public static void main(String []args) {
    try {
      URL url = new URL(args[0]);
      URLConnection urlc = url.openConnection();
      byte []data = new byte[40];
      InputStream is = urlc.getInputStream();
      int lus = is.read(data);
      while (lus>0) {
        for (int i=0; i<lus; i++)
          System.out.print(hexArray[0xff & data[i]]);
        System.out.println();
        lus = is.read(data);
      }
      is.close();
    } catch(Exception e) {
      e.printStackTrace(); System.exit(1);
    }
  }
}
```

```
public class TestURL4 {
  public static void main(String []args) {
    try {
      URL url = new URL(args[0]);
      URLConnection urlc = url.openConnection();
      Object o = urlc.getContent();
      System.out.println(o.getClass());
    } catch(Exception e) {
      e.printStackTrace();
      System.exit(1);
    }
  }
}
```