3.2.3. LA CLASE URL

La clase URL (Uniform Resource Locator) representa un puntero a un recurso en la Web. Un recurso puede ser algo tan simple como un fichero o un directorio, o puede ser una referencia a un objeto más complicado, como una consulta a una base de datos o a un motor de búsqueda.

En general una URL que localiza recursos empleando el protocolo HTTP se divide en varias

partes: http://hostf[:.puerto][/nombredelpathdelservidor][?argumentos], las partes encerradas entre corchetes son opcionales:

- **host:** Es el nombre de la maquina en la que reside el recurso.

- **[:puerto]:** Número de puerto en el que el servidor escucha las peticiones. Este parámetro es opcional y si no se indica se considera el puerto defecto. Para el protocolo HTTP es el 80.

- **[/directoriodelservidor]:** Es el path o directorio donde se encuentra el recurso en el sistema de ficheros del servidor. Sino se indica se proporciona la página por defecto del servidor web.

- **[?argumentos]:** Parámetros que se envían al servidor. Por ejemplo, cuando realizamos una consulta se pueden enviar parámetros a un fichero PHP para procesarla.

La clase URL contiene varios constructores, algunos son:

CONSTRUCTOR	MISIÓN
URL (String url)	Crea un objeto URL a partir del String indicado en *url*.
URL(String protocolo, String host, String fichero)	Crea un objeto URL a partir de los parámetros *protocolo*, *host* y *fichero* (o directorio).
URL(String protocolo, String host, int puerto, String fichero)	Crea un objeto URL en el que se especifica el *protocolo*, *host*, *puerto* y *fichero* (o directorio) representados mediante String.
URL(URL contexto, String especificación)	Crea un objeto URL analizando la especificación dada dentro de un contexto específico.

Estos pueden lanzar la excepción MalformedURLException si la URL está mal construida, no se hace ninguna verificación de que realmente exista la maquina o el recurso en la red.

Algunos de los métodos de la clase URL son los siguientes:

MÉTODOS	MISIÓN
String getAuthority ()	Obtiene la autoridad del objeto URL.
int getDefaultPort()	Devuelve el puerto asociado por defecto al objeto URL.
int getPort()	Devuelve el número de puerto de la URL, -1 si no se indica.
String getHost()	Devuelve el nombre de la máquina.
String getQuery()	Devuelve la cadena que se envía a una página para ser procesada (es lo que sigue al signo? de una URL).
String getPath()	Devuelve una cadena con la ruta hacia el fichero desde el servidor y el nombre completo del fichero.
String getFile()	Devuelve lo mismo que *getPath ()*, además de la concatenación del valor de *getQuery()* si lo hubiese. Si no hay una porción consulta, este método y *getPath()* devolverán los mismos resultados.
String getProtocol()	Devuelve el nombre del protocol asociado al objeto URL.
String getUserInfo()	Devuelve la parte con los datos del usuario o nulo si no existe.
InputStream openStream()	Devuelve un **InputStream** del que podremos leer el contenido del recurso que identifica la URL.
URLConnection openConnection()	Devuelve un objeto **URLConnection** que nos permite abrir una conexión con el recurso y realizar operaciones de lectura y escritura sobre él.

El siguiente ejemplo muestra el usa de los constructores definidos anteriormente; el método Visualizar() muestra información de la URL usando los métodos de la tabla anterior:

```
import java.net.*;
        public class  Ejemplo1URL{
        public  static  void main(String[]  args) {
        URL  url;
        try {
                System.out.println("Constructor  simple  para  una  URL:");
                url  =  new URL("http://docs.oracle.com/");
                Visualizar(ur1);

                System.out.println("Otro  constructor  simple para  una  URL:");
                url = new URL("http://localhost/PFC/gest/cli_gestion.php?S=3");
                Visualizar(ur1);

                System.out.println("Const.  para  protocolo        +URL       + directorio:");
                url = new URL("http",  "docs.oracle.com",  "ijavase/10");
                Visualizar(ur1);
                System.out.println("Constructor para  protocolo  +  URL +  puerto +   +directorio:");
                url =  new  URL("http",  "localhost",   8084,"/WebApp/Controlador?accion=modificar");
                Visualizar(ur1);
```

```
                System. out
                        .println("Constructor para un objeto URL en un contexto:");
                URL urlBase = new URL("https://docs.oracle.com/");
                url = new URL(urlBase,"/javase/10/docs/api/javainet/URL.html");
                Visualizar(url);
} catch (NialformedURLException e) { System.out.println(e);}
}// main
private static void Visualizar(URL url){
        System.out.println("\tURL completa: "          + url.toString());
        System.out.println("\tgetProtocol(): " + url.getProtocol());
        System.out.println("\tgetHost(): "     + url.getHost());
        System.out.println("\tgetPort(): "     + url.getPort());
        System.out.println("\tgetFile(): "     + url.getFile());
        System.out.println("\tgetUserInfo(): " + url.getUserInfo());
        System.out.println("\tgetPath(): "     + url.getPath());
        System.out.println("\tgetAuthority(): "          + url.getAuthority());
        System.out.println("\tgetQuery(): "    + url.getQuery());
        System.out.println("\tgetDefaultPort(): "+ url.getDefaultPort());
        System.out
                .println("=====================================");
                }//
}// Ejemplo1URL
```

La salida generada es la siguiente:

```
        Constructor simple para una URL:
        URL completa: http://docs.oracle.com/
        getProtocol(): http
        getHost(): docs.oracle.com
        getPort():-1
        getFile():/
        getUserInfo(): null
        getPath():          /
        getAuthority(): docs.oracle.com
        getQuery(): null
```

getDefaultPort(): 80

==========

Otro constructor simple para una URL:

URL completa: http://localhost/PFC/gest/cli_gestion.php?S=3

getProtocol(): http

getHost(): localhost

getPort():-1

getFile(): /PFC/gest/cli_gestion.php?S=3

getUserInfo(): null

getPath(): /PFC/gest/cli_gestion.php

getAuthority(): localhost

getQuery(): S=3

getDefaultPort(): 80

==========

Const. para protocolo +URL + directorio:

URL completa: http://docs.oracle.com/javase/10

getProtocol(): http

getHost(): docs.oracle.com

getPort():-1

getFile(): /javase/10

getUserInfo(): null

getPath(): /javase/10

getAuthority(): docs.oracle.com

getQuery(): null

getDefaultPort(): 80

==========

Constructor para protocolo + URL + puerto + directorio:

URL completa: http://localhost:8084/WebApp/Controlador?accion-modificar

getProtocol(): http

getHost(): localhost

getPort():8084

getFile():/WebApp/Controlador?accion-modificar

getUserInfo(): null

getPath(): /WebApp/Controlador

getAuthority(): localhost:8084

getQuery(): accion=modificar getDefaultPort(): 80

==========

Constructor para un objeto URL en un contexto:

URL completa:

https://docs.oracle.com/javase/10/docs/api/java/net/URL.html

getProtocol(): https

getHost(): docs.oracle.com

getPort():-1

getFile(): /javase/10/docs/api/java/net/URL.html

getUserInfo(): null

getPath(): /javase/10/docs/api/java/net/URL.html

getAuthority(): docs.oracle.com

getQuery(): null

getDefaultPort(): 443

==========

El siguiente ejemplo crea un objeto URL a la direccion http://www.elaltozano.es, abre una conexión con el creando un objeto InputStream y lo utiliza como flujo de entrada para leer los datos de la página inicial del sitio; al ejecutar el programa se muestra en pantalla el código HTML de la página inicial del sitio:

```
import java.net.*;
import java.io.*;

public class Ejemplo2URL{
public static void main(String[] args) {
                    URL url=null;
try{
url = new URL("http://www.elaltozano.es");
}catch (MalformedURLException e) e.printStackTrace();)

BufferedReader in;
try{
InputStream inputStream = url.openStream();
in = new BufferedReader(newInputStreamReader(inputStream));
String inputLine;
while((inputLine= in.readLine())!=, null
System.out.println(inputLine);
```

```
in.close();

} catch     (I0Exception  e) {e.printStackTrace();}

}//

)//Ejemplo2URL
```

3.2.4. LA CLASE URLConnection

Una vez que tenemos un objeto de la clase URL, si se invoca al método openConnection()para realizar la comunicación con el objeto y la conexión se establece satisfactoriamente, entonces tenemos una instancia de un objeto de la clase URLConnection:

```
URL  url = new URL("http://www.elaltozano.es");

URLConnection ur1Con= url.openConnection();
```

La clase URLConnection es una clase abstracta que contiene métodos que permiten la comunicación entre la aplicación y una URL. Para conseguir un objeto de este tipo se invoca al método openConnectionO, con ello obtenemos una conexión al objeto URL referenciado. Las instancias de esta clase se pueden utilizar tanto para leer como para escribir al recurso referenciado por la URL. Puede lanzar la excepción I0Exception.

Algunos de los métodos de esta clase son:

MÉTODOS	MISIÓN
InputStream getInputStream()	Devuelve un objeto **InputStream** para leer datos de esta conexión.
OutputStream getOutputStream()	Devuelve un objeto **OutputStream** para escribir datos en esta conexión.
void setDoInput (boolean b)	Permite que el usuario reciba datos desde la URL si el parámetro *b* es *true* (por defecto está establecidoa *true*)
void setDoOutput((boolean b)	Permite que el usuario envíe datos si el parámetro *b* es *true* (no está establecido al principio)
void connect()	Abre una conexión al recurso remoto si tal conexión no se ha establecido ya.
int getContentLength()	Devuelve el valor del campo de cabecera *content-lenght* o -1 si no está definido.
String getContentType()	Devuelve el valor del campo de cabecera *content-type* o null si no está definido.
long getDate()	Devuelve el valor del campo de cabecera *date* o 0 si no está definido.
long getLastModified()	Devuelve el valor del campo de cabecera *last-modified*
String getHeaderField(int n)	Devuelve el valor del enésimo campo de cabecera especificado o null si no está definido.
Map< String, List<String> > getHeaderFields()	Devuelve una estructura Map (estructura de Java que nos permite almacenar pares clave/valor.) con los campos de cabecera. Las claves son cadenas que representan los nombres de los campos de cabecera y los valores son cadenas que representan los valores de los campos correspondientes.
URL getURL()	Devuelve la dirección URL.

El siguiente ejemplo crea un objeto URL a la dirección http://www.elaltozano.es, se invoca al método openConnection() del objeto para crear una conexión y se obtiene un URLConnection. Después se abre un stream de entrada sobre esa conexión mediante el método getInputStream(). Al ejecutar el programa se muestra la misma salida que en el ejemplo anterior; sin embargo, este programa crea una conexión con el recurso representado por la URL y el anterior abre directamente un stream desde la URL:

```
import java.net.*;
import java.io.*;

        public class Ejemplo1urlCon {
                public static void main(String[] args) {
                URL url=null;
                URLConnection urlCon=null;
                try{
                url = new URL("http://www.elaltozano.es");
                urlCon= url.openConnection();

                BufferedReader in;
                InputStream inputStream = urlCon.getInputStream();
                in = new BufferedReader(new
                                InputStreamReader(inputStream));
                String inputLine;
                while((inputLine= in.readLine())!= null)
                        System.out.println(inputLine);
                in.close();
        }
        catch (MalformedURLException e) {e.printStackTrace();}
        catch (IOException e) {e.printStackTrace();}
}//
}// Ejemplo1urlCon
```

Gran cantidad de páginas HTML contienen formularios a través de los cuales podemos solicitar información a un servidor rellenando los campos requeridos y pulsando al botón de envio. El servidor recibe la petición, la procesa y envía los datos solicitados al cliente normalmente en formato HTML. Por ejemplo, tenemos una página HTML que contiene un formulario con dos campos de entrada y un botón. En el atributo "action" se indica el tipo de acción que va a realizar el formulario, en este caso los datos se envían a un script PHP de nombre vernombre.php; con method=post indicamos la forma en que se envia el formulario:

```
<html>
<body>
<form action="vernombre.php" method="post" >
<p>Escribe to nombre:
<input name="nombre" type="text" size="15"></p>
<p>Escribe tus apellidos:
<input name="apellidos" type="text" size="15"></p>
                    <input type="submit" name="ver" value="Ver">
</form>
</body>
</html>
```

El script PHP que recibe los datos del formulario es el siguiente:

```
<?php
$nom=$_POST[“nombre"];
Sape=$_POST["apellidos"];
echo "El nombre recibido es:          $nom, y “;
echo "los apellidos son: $ape ";
?>
```

En él se reciben los valores introducidos en los campos nombre y apellidos del formulario, mediante la instrucción $POST[“nontbrecampo”], y se visualizan en la pantalla del navegador mediante la orden echo. La URL para enviar datos al formulario, suponiendo que los ficheros .html y .php residen en la carpeta 2018 del servidor web local sería similar a la siguiente:

http ://localhost/2018/vernombre.php?nombre =Maria&apellidos = Ramos &ver= Ver

Tarea 3.2

Desde Java usando la clase **URLConnection** podemos interactuar con scripts del lado del servidor y podemos enviar valores a los campos del script sin necesidad de abrir un formulario HTML, será necesario escribir en la URL para dar los datos al script. Nuestro programa tendrá que hacer lo siguiente:

- Crear el objeto URL al script con el que va a interactuar. Por ejemplo, en nuestra maquina local tenemos instalado un servidor web Apache y dentro de htdocs tenemos la carpeta 2018 con el script PHP vernombre.php, la URL seria la siguiente: URL url=new URL(http://localhost/2018/vernombre.php).

- Abrir una conexión con la URL, es decir obtener el objeto URLConnection: URLConnection conexion = urLopenConnection().

- Configurar la conexion para que se puedan enviar datos usando el método setDoOutput(): conexion.setDoOutput(true).

■ Obtener un stream de salida sobre la conexión: PrintWriter output = new PrintWriter(conexion.getOutputStream0).

■ Escribir en el stream de salida, en este caso mandamos una cadena con los datos que necesita el script: output.write(cadena). La cadena tiene el siguiente formato: parametro=valor, si el script recibe varios parametros seria: parametro1=valor1 &parametro2=valor2&parametro3=valor3, y así sucesivamente.

■ Cerrar el stream de salida: output.close().

Normalmente cuando se pasa información a algún script PHP, este realiza alguna acción y después envía la información de vuelta por la misma URL. Por tanto si queremos ver lo que devuelve será necesario leer desde la URL. Para ello se abre un stream de entrada sobre esa conexión mediante el método getInputStream(): BufferedReader reader = new BufferedReader (new InputStreamReader(conexion.getInputStream())); y después se realiza la lectura para obtener los resultados devueltos por el script.