ID2212 Network Programming with Java
Lecture 7

# Working with Web Resources and URL Connections

Leif Lindbäck, Vladimir Vlassov

KTH/ICT/SCS

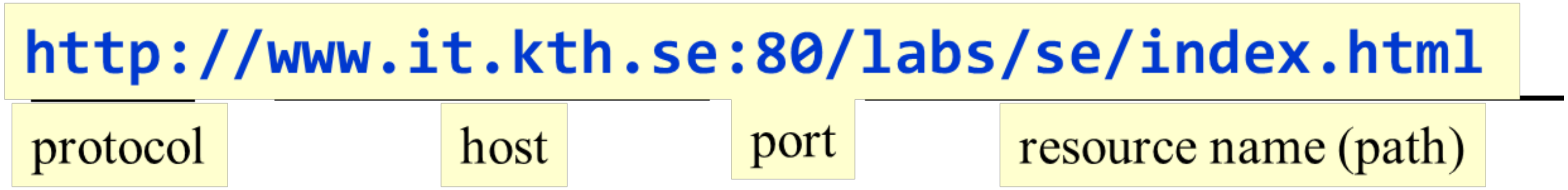
HT 2015

# Outline

- Accessing Web Resources in Java
  - Locating Web resources: URL
  - Communicating with HTTP servers: URL connections
    - Downloading and uploading data on the Web
  - Retrieving Web resources pointed to by URLs: content handlers
    - Images, audio, HTML documents, classes, files
- Developing custom protocol and content handlers.

# Locating Web Resources: `java.net.URL`

- `java.net.URL` **represents an URL - Uniform Resource Locator of a Web resource (service)**
  - **A URL object is used to locate and to grab Web resources.**
    - **The JDK provides implementations for many different protocols, for wxample HTTP and FTP.**
- **A URL with the http scheme:**



`http://www.it.kth.se:80/labs/se/index.html`

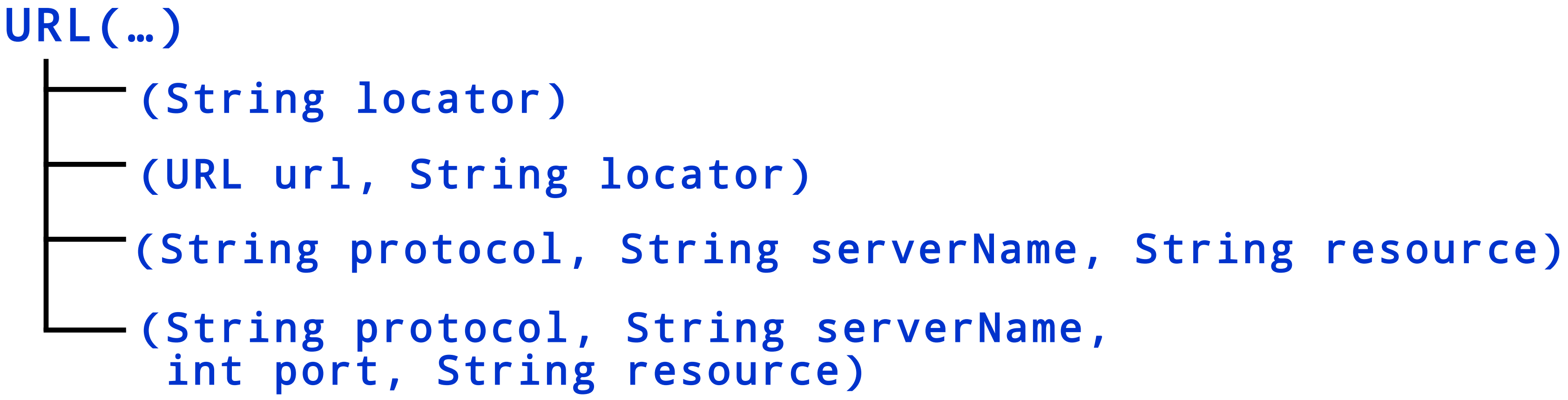
protocol

host

port

resource name (path)

# URL Constructors



URL(…)

(String locator)

(URL url, String locator)

(String protocol, String serverName, String resource)

(String protocol, String serverName,
 int port, String resource)

```
 try {
    URL url1 =
      new URL(“http://www.ora.com/info/java/index.html“);
    // create an absolute URL from a base and a
    // relative URL:
    URL url2 = new URL(url1, “bibliography.html“);
} catch (MalformedURLException e) {
   e.printStackTrace();
}
```

# URL Stream Handler

- A stream protocol handler that knows how to make a connection for a particular protocol type, such as http, ftp.
- A stream handler can be found by a `URL` object the following two ways.
  - If `URLStreamHandlerFactory` has been set by a call to `URL.setURLStreamHandlerFactory`, call its method `createURLStreamHandler`
  - Else, try to load the class `sun.net.www.protocol.<protocol>.Handler`

# Communicating with a Web Server: `java.net.URLConnection`

- The `URLConnection` class represents a communication link to the resource.

```
URL url =
    new URL(“http://www.it.kth.se/index.html”);
URLConnection urlc = url.openConnection();
```

- The server is contacted only when needed, for example when the content or a header field is read:

```
myUrlConnection.getContent()
```

`myUrlConnection.getContentLength()`, etc.

# Input and Output Streams of URLConnection

- URL connection provides input and output streams
  - Input stream for downloading the resource contents, e.g. classes, images.

    ```
    URLConnection urlc =url.openConnection();
    InputStream in = urlc.getInputStream();
    ```

  - Output stream for uploading data to the server at the corresponding URL, e.g. posting query to a CGI script or a servlet

    ```
    URLConnection urlc =url.openConnection();
    OutpuStream o = urlc.getOutputStream();
    ```

# Retrieving Resource Information and Content (HTTP)

- Methods of `URLConnection` to get information about the resource and its content:

  `getContentLength`

  `getContentType`

  `getContent`

  `getContentEncoding`

  `getDate`

  `getExpiration`

  `getLastModified`

  - The information is obtained via the `GET` request.
  - The server sends a MIME header and resource data in reply.

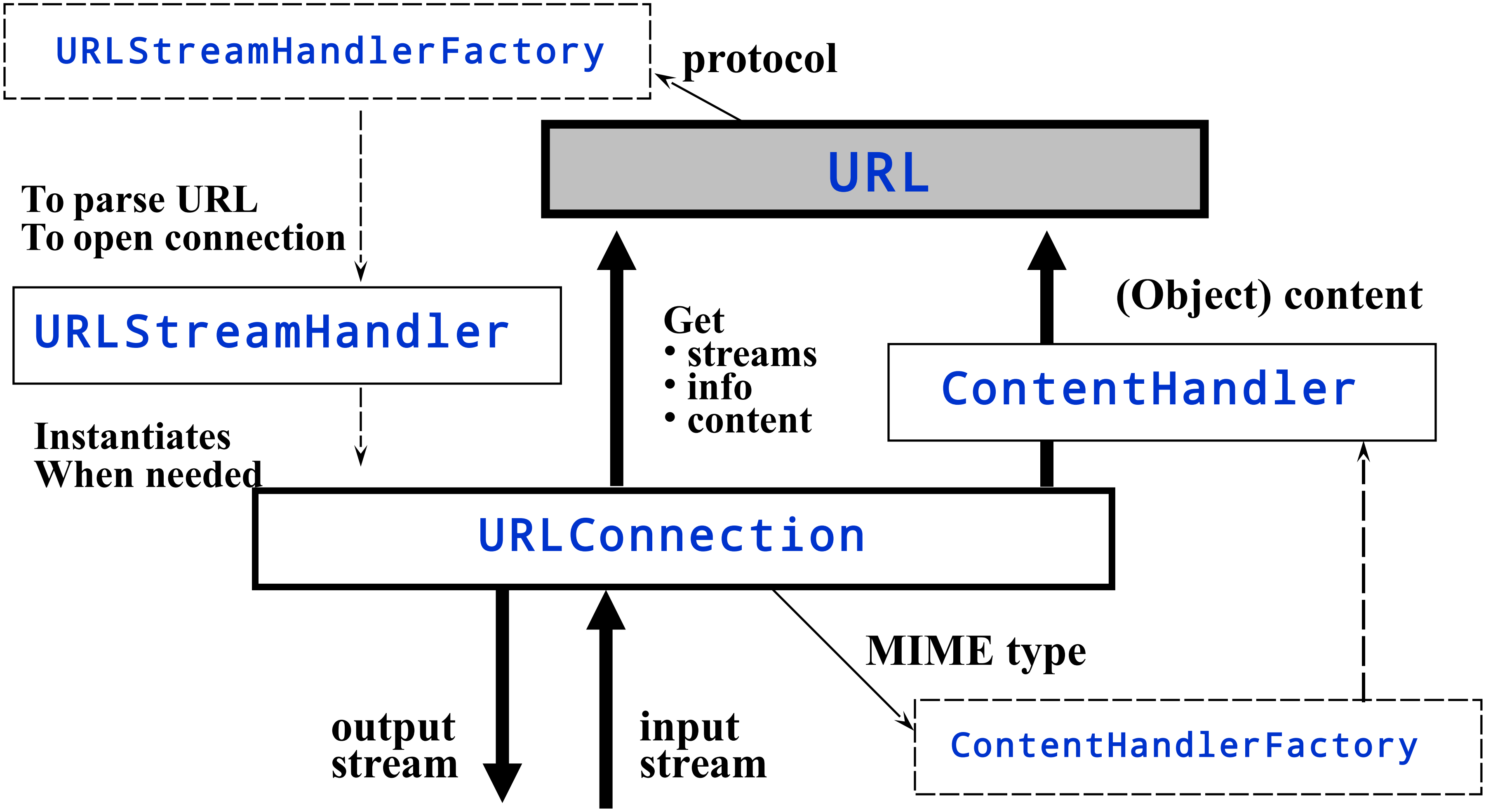
# Handling downloaded content: `java.net.ContentHandler`

- A ContentHandler has a `getContent` method that reads data from the input stream of the given URL connection and converts it into a Java object.
- The handler is associated with one or several MIME types that it can handle.
- The handler can support caching of content.

# Looking for a Content Handler

- A content handler can be found by a `URLConnection` object the following two ways.
  - If a `ContentHandlerFactory` has been set by a call to `URLConnection.setContentHandlerFactory`, call its method `createContentHandler(String mimeType)`, otherwise:
  - Try to load the class `sun.net.www.content.<mimeType>.<mimeSubType>`

# Architecture of URL Related Classes



# The Shorthand Method `url.getContent()`

1. Creates a `URLStreamHandler` instance
2. Calls `URLStreamHandler.openConnection(URL)`
   - The stream handler creates a URL connection
3. Calls `URLConnection.getContent()`
   - The connection connects to server
4. Calls `URLConnection.getContentHandler()`
   - The connection creates a content handler for a given MIME type such as “text/html”
5. Calls `ContentHandler.getContent()`
   - The handler converts downloaded data into a Java object

# An Example, GetHttpResource

```
public HttpResource(String url) throws
  MalformedURLException {
  this.resourceUrl = new URL(url);
}

public void loadResource() throws IOException {
  System.out.println("MIME: " +
           resourceUrl.openConnection().getContentType());
  Object resource = resourceUrl.getContent();
  if (resource instanceof ImageProducer) {
    // The content handler returns an ImageProducer if
    // the resource is an image. If so, show the image.
} else if (resource instanceof InputStream) {
    // Print stream content to System.out
}
```

# Developing a Custom Protocol, *Ping*

- Steps:
  - Develop a `URLStreamHandler` subclass for the protocol
  - Develop a `URLStreamHandlerFactory` implementation for the protocol
  - Develop a `URLConnection` subclass for the protocol
  - Optional, develop a `ContentHandler` subclass and a `ContentHandlerFactory` implementation.