Introducción a la JMF

Servidores de Información Multimedia
2º Ingeniero Técnico de Telecomunicación – Imagen y Sonido

Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid

2 Índice

- Presentación de la JMF
- Modelo de procesamiento de la JMF
- Extensibilidad de la JMF
- Entrada en JMF
- Reproduciendo la información
- Procesando información
- Guardando la información
- Aplicación de ejemplo

3 Presentación de la JMF

- La **Java Media Framework** (JMF) es un API para la manipulación y procesamiento de medios en Java.
- Permite la captación de medios (de micrófono, red...), su procesamiento (multiplexación, codificación...), su reproducción, su almacenamiento y su difusión.
- Una de las principales características de los medios es su dependencia con la variable tiempo ("time-based media") . La JMF tiene que cumplir requisitos temporales en el manejo de medios.
- Las APIs de la JMF pueden usarse para realizar aplicaciones y Applets Java (veremos varios ejemplos a lo largo de la asignatura)

4 Presentación de la JMF (cont)

- Versiones:
  - JMF 1.0 API
    - Daba soporte a la presentación (reproducción) de medios.
  - JMF 2.0 API
    - Da soporte adicional para la captación de medios (de dispositivos de captura o red) y para su almacenamiento (en disco o difusión a través de la red),
    - Permite contralar el tipo de procesamiento que se realiza durante la reproducción,
    - Tiene soporte para plug-ins mediante un API que permite extender la JMF
  - JMF 2.1.1
    - No hay cambios significativos en el API (quizás lo más significativo en torno a RTP), hay una actualización en la implementación de Sun

5 Presentación de la JMF (cont)

- Algunos paquetes Java definidos en la JMF:
  - javax.media,
  - javax.media.bean.playerbean,
  - javax.media.control,
  - javax.media.datasink,
  - javax.media.format,
  - javax.media.protocol,
  - javax.media.renderer,
  - javax.media.rtp,
  - javax.media.rtp.event,
  - javax.media.rtp.rtcp,
  - javax.media.util

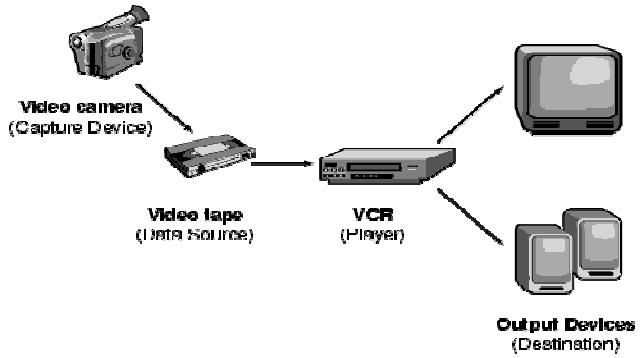
6 Presentación de la JMF (cont)

- JMF soporta la transmisión y recuperación de medios por la red mediante RTP.
- Los APIs para RTP no son obligatorios en la especificación de la JMF
- Sin embargo la implementación de referencia de Sun e IBM si que soporta por completo el API de RTP (es la implementación que vamos a utilizar)
- La implementación de referencia de la JMF tiene versiones para Windows, Solaris, Linux, además de una versión **pure-Java** (que funciona con cualquier JVM)
- Es preferible instalar una versión específica para el SO pues da unos rendimientos mejores.

- Para obtener la implementación de referencia de la JMF:
  - http://java.sun.com/products/java-media/jmf/
- Para perder el miedo, veamos unas **demos**:
  - http://java.sun.com/products/java-media/jmf/2.1.1/samples/index.html

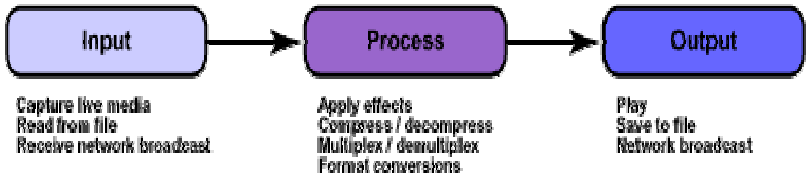
7 Modelo de procesamiento de la JMF

- La figura muestra el modelo de componentes usado por la JMF



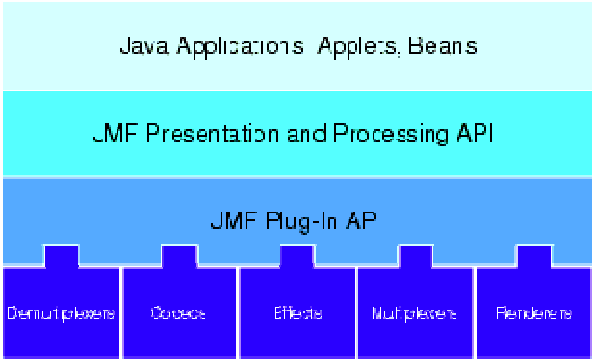
8 Modelo de procesamiento de la JMF (cont)

- Modelo general:



- Ejemplo:

9 Extensibilidad de la JMF



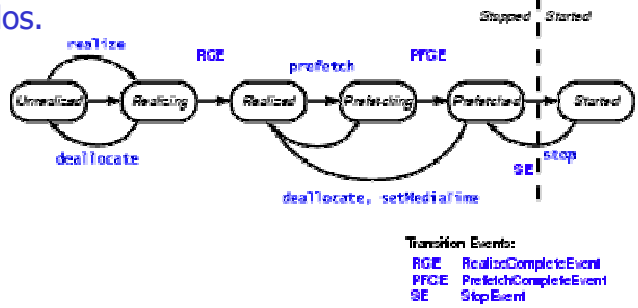
- La JMF soporta Plug-ins como se muestra en la figura:

10 Entrada en JMF

- La entrada de datos (Input) en la JMF se direcciona normalmente por un objeto *javax.media.MediaLocator*.
- Un *MediaLocator* es parecido a una *URL* – identifica de forma única un recurso en la red. La ventaja es que el *MediaLocator* no tiene por qué tener definido ningún manejador (handler) para su protocolo de cara a poder crearse.
- Ejemplo: MediaLocator para una sesión RTP:
  - rtp://address:port[:ssrc]/content-type/[ttl]
- De un *MediaLocator* se crea una instancia de *javax.media.protocol.DataSource* que representa la entrada de datos para el resto de componentes en la arquitectura: *Player*s, *Processor*s y *DataSink*s.
- Nota: Un dispositivo de captura también tiene asociado un *MediaLocator*

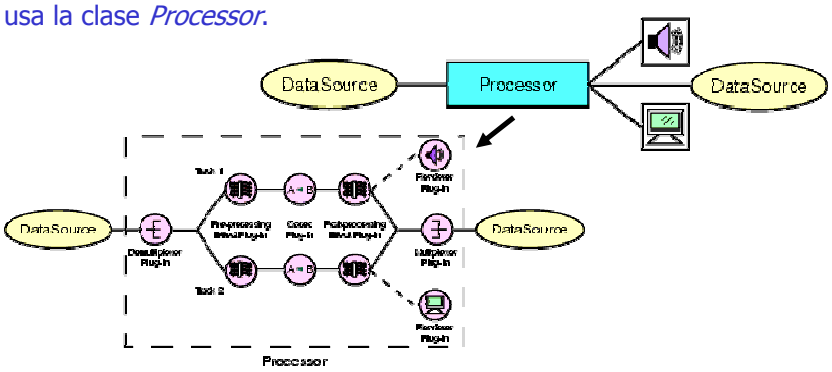
11 Reproduciendo la información

- Se utiliza un *Player*
- El *Player* solo puede presentar/reproducir en pantalla (o altavoces), no puede guardar en fichero.
- Para crearlo se necesita utilizar una clase que hace de factoría para los *Players* que es la clase *Manager*.
- Un *Player* tiene asociado un conjunto de estados.
- No todas las operaciones están permitidas en todos los estados.



12 Procesando información

- Si se quiere tener un control total sobre el procesamiento de los medios (por ejemplo controlar el tipo de codec que se utiliza), o si se quiere poder mandar la información a otro destino distinto de la pantalla y altavoces se usa la clase *Processor*.



- Los *Processors* también se construyen a través del *Manager*.
- También pasan por diferentes estados (se añade un estado de configuración al *Player*)

13 Guardando la información

- Para guardar la información en disco o mandarla a través de la red se usa un *DataSink*.
- Los *DataSink* se construyen a partir de la salida de los *Processor*.
- El destino se especifica como un *MediaLocator.*
- Un destino que realice la difusión por la red mediante RTP no será más que el identificado por un *MediaLocator* como:
  - rtp://address:port[:ssrc]/content-type/[ttl]

14 Aplicación de ejemplo

- El SimplePlayerApplet.java es uno de los ejemplos que vienen con la JMF.
- Veamos su aspecto:

15 Aplicación de ejemplo (cont)

- Paquetes que usa:

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.lang.String;
import java.net.URL;
import java.net.MalformedURLException;
import java.io.IOException;
import java.util.Properties;
import javax.media.*;
```


16 Aplicación de ejemplo (cont)

```
public class SimplePlayerApplet extends Applet implements
   ControllerListener {
    // media Player
    Player player = null;
    // component in which video is playing
    Component visualComponent = null;
    // controls gain, position, start, stop
    Component controlComponent = null;
    // displays progress during download
    Component progressBar = null;
    boolean firstTime = true;
    long CachingSize = 0L;
    Panel panel = null;
    int controlPanelHeight = 0;
    int videoWidth = 0;
    int videoHeight = 0;
```

17 Aplicación de ejemplo (cont)

```
public void init() {
    panel = new Panel();
    panel.setLayout( null );
    add(panel);
    panel.setBounds(0, 0, 320, 240);
    // input file name from html param
    String mediaFile = null;
    // URL for our media file
    MediaLocator mrl = null;
    URL url = null;
    try {
        url = new URL(getDocumentBase(), mediaFile);
        mrl = new MediaLocator(url.toExternalForm());
    } catch (MalformedURLException mue) { }
    try {
        player = Manager.createPlayer(mrl);
    } catch (NoPlayerException e) {    }
    player.addControllerListener(this);
    } catch (Exception e) { }
}
```


18 Aplicación de ejemplo (cont)

```
public void start() {
   //$ System.out.println("Applet.start() is called");
        // Call start() to prefetch and start the player.
        if (player != null)
       player.start();
    }

public void stop() {
   //$ System.out.println("Applet.stop() is called");
        if (player != null) {
            player.stop();
            player.deallocate();
        }
    }
```

19 Aplicación de ejemplo (cont)

```
public synchronized void controllerUpdate (ControllerEvent event) {
    // If we're getting messages from a dead player,
    // just leave
    if (player == null)
        return;

    // When the player is Realized, get the visual
    // and control components and add them to the Applet
    if (event instanceof RealizeCompleteEvent) {
      if (progressBar != null) {
                panel.remove(progressBar);
                progressBar = null; }
      int width = 320;
      int height = 0;
      if (controlComponent == null)
                if (( controlComponent =
                          player.getControlPanelComponent()) != null) {
                controlPanelHeight = controlComponent.getPreferredSize().height;
                panel.add(controlComponent); height += controlPanelHeight; }
...
```


Cuestiones de repaso

- Explique qué es un time-based media y su relación con JMF
- ¿Que versión de JMF: 1.0., 2.0 o 2.1.1 introduce el concepto de plugin?
- ¿Cuál de los siguientes sistemas operativos no tiene implementación de JMF? Solaris, Windows, Linux o QNX
- ¿Cuál de las siguientes empresas: IBM, Sun tienen una implementación propia de JMF?
- Relacione los siguientes elementos: *video camera*, *video tape*, *vcr*, *output device* con sus correspondencias en JMF
- Enumere cuáles son los cinco tipos de plugins que existen en JMF
- Defina qué es un *MediaLocator* y diga la relación que mantiene con un *DataSource*
- Explique cual es la diferencia funcional existente entre un *Player*, un *Processor* y un *DataSink*

Cuestiones de repaso (cont)

- Diga a grosso modo cuáles son los 6 estados en los cuales puede estar un *Player*
- Dibuje la estructura interna de un *Processor* señalando sus principales bloques constructivos
- Explique qué papel juega el *MediaLocator* a la hora de almacenar información en disco.
- ¿Cuál es principal paquete Java en el que encontraremos el Java Media Framework: javax.jmf, java.media o javax.media?
- Explique (con código) el papel que juegan las clases *MediaLocator* y *Manager* en la creación de un *Player*
- Explique (con código) cómo se reciben los eventos que nos envía un *Player*
- Explique para qué sirven los métodos *start*(). *stop*() y *deallocate*() de la clase *Player*

Cuestiones de repaso (cont)

- Explique por qué en el ejemplo visto en clase, no se llama directamente al método *getControlPanelComponent* antes de arrancar el *Player.* ¿Por qué se llama tras haber recibido un evento de tipo *Realize*?

Autoría

- Mario Muñoz Organero
- Pablo Basanta

  + preguntas de control