JMF per applicazioni multimediali in Java

Corso di ***Applicazioni Telematiche***

A.A. 2006-07 – Lezione n. 9

Prof. Roberto Canonico

Lezione a cura di Ing. Generoso Paolillo

Università degli Studi di Napoli Federico II

Facoltà di Ingegneria

Java Media Framework (JMF)

Java Media Framework (JMF) è un package opzionale di Java che permette di utilizzare contenuti multimediali all'interno di applicazioni e applet Java.

- JMF permette di catturare, riprodurre, trasmettere e decodificare più formati multimediali
- estende le funzionalità della piattaforma J2SE
- offre agli sviluppatori multimediali un insieme di strumenti per sviluppare applicazioni scalabili e indipendenti dalla piattaforma.

Obiettivi di progetto di JMF

JMF ha una architettura che permette l'accesso diretto ai dati multimediali e da la possibilità di essere personalizzato ed esteso. JMF è progettato per:

- Supportare la cattura di dati multimediali
- Sviluppare applicazioni Java per lo streaming multimediale
- Permettere a sviluppatori esperti di implementare soluzioni personalizzate basate sulle API esistenti e di integrare facilmente nuove funzionalità con l'infrastruttura esistente
- Permettere l'accesso ai dati grezzi
- Permettere lo sviluppo di componenti personalizzabili (JMF *plug-ins)* come ad esempio demultiplexers, codecs, effects processors, multiplexers, e renderers.

3

JMF RTP APIs

I principali package che compongono le API JMF sono:

- ***javax.media***: contiene le principali classi di JMF
- ***javax.media.rtp****:* contiene le classi per accedere ai campi del protocollo RTP/RTCP
- ***javax.media.control****:* permette di leggere e modificare parametri quali: bit rate, frame rate, lunghezza del buffer di ricezione, e altri parametri del processo di encoding.
- ***javax.media.format****: per la descrizione dei formati supportati*

4

Lavorare con i *time-based media*

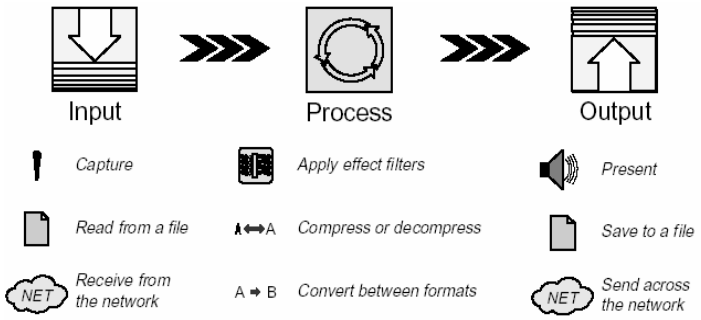
Ogni dato che cambia significativamente nel tempo può essere caratterizzato come time-based media:

es: audio clips, MIDI sequences, movie clips, e animazioni

Tali media possono essere ottenuti da una varietà di sorgenti:

es: file locali o remoti, camere, microfoni e trasmissioni broadcast

Modello di elaborazione di tali dati:



5

Streaming Media

Content type

Il formato nel quale è memorizzato il dato multimediale è detto content type.

Media streams

- Un media stream è un dato multimediale ottenuto da un file locale, acquisito dalla rete o catturato da una camera o da un microfono.
- I media stream contengono spesso più canali di dati detti tracks. Media streams che contengono più tracce vengono detti multiplexed o complex media streams.
- Con *Demultiplexing* viene indicato il processo che permette di estrarre le singole tracce dal complexed media stream. Il tipo di traccia indica il tipo di dato che contiene, audio o video.

6

Streaming Media (cont.)

Un media stream può essere identificato dalla sua posizione e dal protocollo usato per accedervi. Alcuni esempi:

- ***URL*** (Uniform Resource Locator) rappresenta un puntatore ad una risorsa (multimediale in questo caso) sia essa locale (file:/D:/movie/test.mov) o remota (http://WebSite/darkcity.mov).
- ***MediaLocator*****:** offre le stesse funzionalità della classe *URL* ed è specifico per contenuti multimediali

- I media stream possono essere classificati in base alla modalità con la quale i dati sono consegnati:
  - **Pull**: il trasferimento dati è iniziato e controllato dal lato client. Ad esempio il protocolli HTTP è un protocollo pull.
  - **Push**: Il server inizia il trasferimento dei dati e controlla il flusso. Per esempio il protocollo RTP (Real-time Transfert Protocol) è un protocollo push usato per lo streaming di dati multimediali.

7

Common Video Formats

La tabella seguente riporta alcune caratteristiche dei più comuni formati multimediali video:

Format	Content Type	Quality	CPU Requirements	Bandwidth Requirements
Cinepak	AVI QuickTime	Medium	Low	High
MPEG-1	MPEG	High	High	High
H.261	AVI RTP	Low	Medium	Medium
H.263	QuickTime AVI RTP	Medium	Medium	Low
JPEG	QuickTime AVI RTP	High	High	High
Indeo	QuickTime AVI	Medium	Medium	Medium

8

Common Audio Formats

La tabella seguente riporta alcune caratteristiche dei più comuni formati multimediali audio:

Format	Content Type	Quality	CPU Requirements	Bandwidth Requirements
PCM	AVI QuickTime WAV	High	Low	High
Mu-Law	AVI QuickTime WAV RTP	Low	Low	High
ADPCM (DVI, IMA4)	AVI QuickTime WAV RTP	Medium	Medium	Medium
MPEG-1	MPEG	High	High	High
MPEG Layer3	MPEG	High	High	Medium
GSM	WAV RTP	Low	Low	Low
G.723.1	WAV RTP	Medium	Medium	Low

9

Media presentation

La maggior parte dei time-based media sono dati audio o video che possono essere presentati attraverso dispositivi di output come altoparlanti o monitor. I media streams possono essere anche spediti ad altre destinazioni, per esempio salvate in file o trasmesse sulla rete. Una destinazione di uscita per dati multimediali viene detta ***data sink***.

Latency

In molti casi, particolarmente quando viene riprodotto un media stream proveniente dalla rete, la presentazione del media stream può non essere immediata. Il tempo che precede la presentazione viene detto *start latency*.

Una presentazione multimediale spesso combina diversi tipi di time-based media in un unica presentazione sincronizzata. Quando la presentazione di più media stream è sincronizzata è fondamentale prendere in considerazione la *start latency* di ogni stream.

10

Media presentation (cont.)

Qualità della presentazione

La qualità di una presentazione multimediale dipende da diversi fattori quali:

- Lo schema di compressione usato
- La capacità elaborativa del sistema che effettua la presentazione
- La banda disponibile (per media stream provenienti dalla rete)

Per ottenere presentazioni video di maggiore qualità, il numero di frame per secondo visualizzate deve avvicinarsi quanto più possibile ai 30 fps. Filmati con frame rate superiori ai 30 frame per secondo sono pressoché indistinguibili dall'occhio umano.

11

Media Processing

In molti casi il media stream prima di essere presentato all'utente viene elaborato. Di solito, una serie di elaborazioni sono necessarie prima della presentazione:

- Se lo stream è multiplexed, le singole tracce sono estratte
- Se le singole tracce sono compresse, vengono decodificate
- Se necessario, le tracce sono convertite in un formato differente
- Effetti di filtraggio possono essere applicati alle tracce decodificate

Demultiplexers e Multiplexers

Un Demultiplexer estrae singole tracce multimediali da media stream multiplexed. Un Multiplexer esegue la funzione inversa.

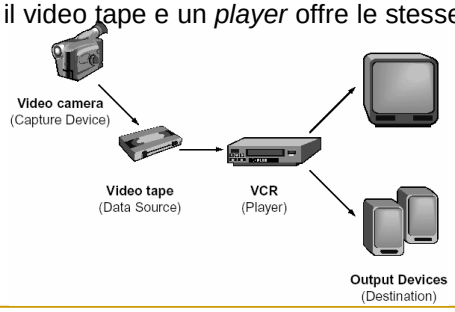
Codecs

Un codec effettua la compressione e decompressione dei dati multimediali. Quando una traccia è codificata (encoded), viene convertita in un formato compresso adatto per la memorizzazzione o la trasmissione; quando viene decodificata (decoded) è convertita in un formato non compresso (raw) adatto per la presentazione.

12

Modello di riferimento

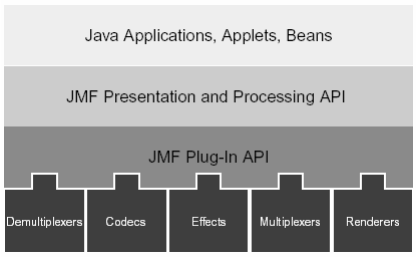
- Dispositivi come un videoregistratore (VCR) offrono un modello familiare per registrare, elaborare, e presentare dei time-based media.
- Riprodurre un filmato con un VCR, equivale a spedire un media stream al VCR inserendo un video tape. Il VCR legge e interpreta i dati della cassetta generando gli opportuni segnali al televisore e agli altoparlanti.
- JMF usa lo stesso modello base. Un *data source* incapsula il media stream così come fa il video tape e un *player* offre le stesse funzionalità del VCR.



13

Modello di riferimento (cont.)

- Un *data source* incapsula sia la posizione del media che il protocollo per consegnare il media, costituisce infatti una astrazione di media protocol-handler.
- Data source e player sono parte integrante delle API di alto livello di JMF per la gestione della cattura, presentazione, ed elaborazione di time-based media. L'architettura di alto livello di JMF è riportata di seguito:



14

Manager

Le API di JMF consistono principalmente di interfacce che definiscono il comportamento e l'interazione di oggetti usati per catturare, elaborare e presentare time-based media. Usando degli oggetti intermediari chiamati *managers*, JMF rende facile l'integrazione di nuove implementazioni di interfacce che possono essere usate con le classi esistenti. JMF usa quattro managers:

- **Manager-** gestisce la costruzione di *Players*, *Processors*, *DataSources*, e *DataSinks*. Questo livello di redirezione permette a nuove implementazioni di essere integrate in JMF. Dalla prospettiva del client, questi oggetti sono creati allo stesso modo sia che l'oggetto richiesto sia implementato di default sia che costituisca una realizzazione personalizzata (custom).
- **PackageManager-** gestisce un registro di package che contengono le classi JMF, come dei custom *Player*, *Processor*,…
- **CaptureDeviceManager-** gestisce un registro dei dispositivi di acquisizione disponibili
- **PlugInManager-** gestisce un registro dei plug-in di componenti di elaborazione quali Multiplexers, Demultiplexers, Codecs, Effects e Renders.

15

Manager (cont.)

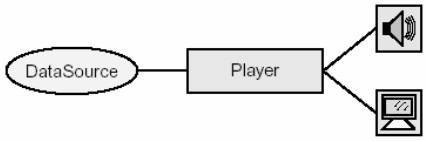
Una generica applicazione basata su JMF utilizza il metodo *create()* del *Manager* per costruire i *Players*, *Processors*, e *DataSinks*.

Se si è interessati ad acquisire dei media da dispositivi di input, è necessario usare la classe *CaptureDeviceManager* per trovare i dispositivi disponibili e accedere alle loro informazioni.

16

Presentazione di time-based media

- Per riprodurre un time-based media come audio e video con JMF bisogna creare un oggetto *Player*.
- Il player viene creato attraverso il metodo createPlayer() su *Manager* passando come parametro o una *URL* (Uniform Resource Locator) o un *MediaLocator* oppure un *DataSource*. La maggior parte dei metodi che possono essere invocati sul Player necessitano che il Player sia in uno stato Realized.



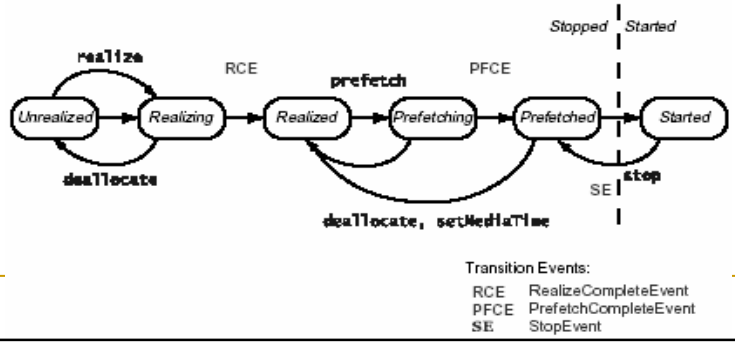
17

Stati di un Player

- Un player si può trovare in uno qualsiasi dei seguenti sei stati:

Unrealized,Realizing, Realized, Prefetching, Prefetched ,Started.

- Un Player istanziato ma senza un media associato si trova in uno stato Unrealized, quando ha acquisito le risorse si porta in uno stato Realized. Quando viene invocato il metodo Prefetch il Player carica il media preparandosi per la riproduzione e transitando nello stato Prefetched. A questo punto il metodo Start fa passare il Plaver nello stato Started.



18

Controllo degli stati di un Player

- L'interfaccia Player eredita dall'interfaccia Controller per cui, se la nostra classe vuole diventare listener degli eventi del Player deve implementare l'interfaccia ControllerListener e registrarsi come listener del Player attraverso il metodo:

  addControllerListener(this)

  eseguito sul Player.
- L'esempio ProvaPlayer.java mostra la creazione di un Player e la gestione degli eventi generati da questo nelle transizioni di stato.

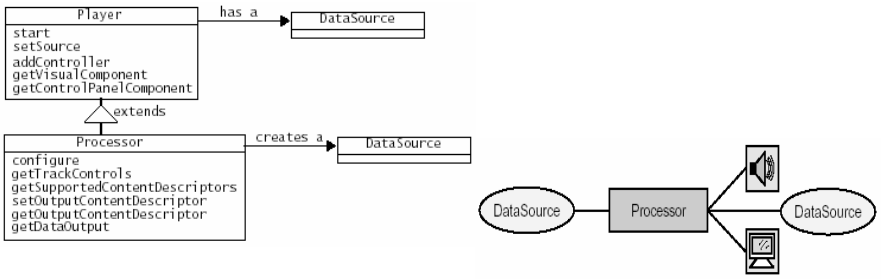
Eseguire il programma passando come parametri di ingresso i seguenti MediaLocator :

- file:/D:/myDirectory/movie.mov
- http://java.sun.com/products/java-media/jmf/2.1.1/samples/samples/media/darkcity.7.160x120.11khz.mov

19

Processor

Un Processor è un Player che prende un DataSource in input, esegue qualche elaborazione definita dall'utente sui dati multimediali, e poi rilascia l'output. A differenza di un Player può spedire i dati di output a un dispositivo di presentazione oppure ad un DataSource. Se i dati sono spediti ad un DataSource, questo può essere usato a sua volta come input per un altro Player o Processor.

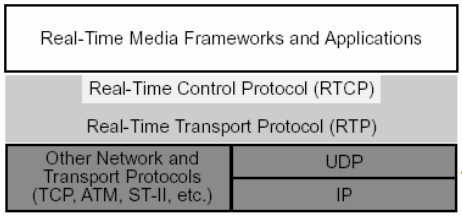


20

Real-Time Media Stream

Real-Time Transport Protocol (RTP)

- RTP offre un servizio di consegna end-to-end per trasmissioni di dati real-time.
- RTP è indipendente dalla rete e dal protocollo di trasporto, sebbene sia spesso utilizzato su UDP.
- RTP può essere usato sia per servizi di rete di tipo unicast che multicast. Con servizi di rete unicast, diversi flussi dati sono spediti dalla sorgente ad ogni destinazione. Con servizi di rete multicast, i dati sono spediti da una sola sorgente ed è la rete che si occupa di consegnarla ai vari destinatari.



21

Real-Time Media Stream (cont.)

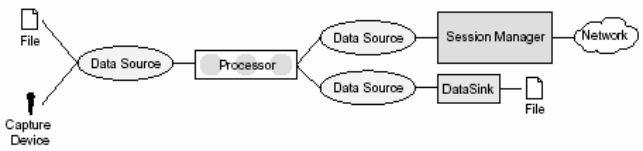
- RTP non offre alcun meccanismo per assicurare una consegna tempestiva dei dati
- Il protocollo di controllo, RTCP, permette però di monitorare la qualità della trasmissione dei dati attraverso dei pacchetti generati dai client che periodicamente riassumono la qualità della propria ricezione.

22

Trasmissione di stream RTP

Per trasmettere uno stream RTP si seguono i seguenti passi:

- Creare un *DataSource* o *MediaLocator* per identificare la sorgente dello stream
- Creare un *Processor* da collegare alla sorgente del punto precedente per produrre in uscita un *DataSource* con un particolare formato compresso RTP (attraverso la configurazione del *Processor*)
- A questo punto sono possibili due soluzioni:
  - Creare un *DataSink* che prenda in input il *DataSource* di uscita dal *Processor* per controllare la trasmissione
  - Creare un *RTPManager* per la trasmissione che permette la trasmissione di stream multipli nella sessione e permette inoltre di monitorare le statistiche associate alla sessione.



23

Trasmissione di stream RTP (cont.)

Un esempio di trasmissione di un flusso multimediale RTP con l'uso di un DataSink è offerto dalla classe VideoTransmit.java.

Essa prende in input come primo parametro una URL del tipo:

- file:/C:/media/speech.mov
- http://mediacentral.com/speech.avi
- vfw://0

Come secondo parametro l'indirizzo IP della macchina destinazione e come terzo parametro il porto destinazione.

Può quindi essere lanciato col comando:

java VideoTransmit file:/C:/media/speech.mov 224.224.224.224 48000

24

Trasmissione di stream RTP (cont.)

In alternativa un esempio di utilizzo dell'RTPManager è proposto nella classe AVTransmit2.java. Essa crea due sessioni sull'indirizzo IP dato come parametro differenziando il porto della seconda sessione di 2 unità.

In input come primo parametro accetta una URL del tipo:

- file:/C:/media/speech.mov
- http://mediacentral.com/speech.avi
- vfw://0

Come secondo parametro l'indirizzo IP della macchina destinazione e come terzo parametro il porto destinazione.

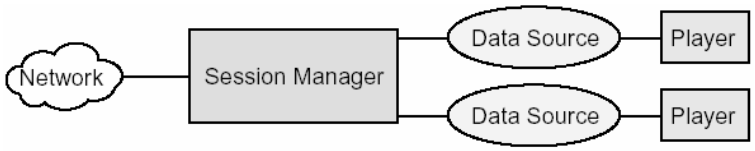
Può quindi essere lanciato col comando:

java AVTransmit2 file:/C:/media/speech.mov 143.225.229.13 48000

25

Ricezione e presentazione di stream RTP

- Per ricevere e riprodurre uno stream multimediale proveniente dalla rete bisogna costruire un *Player* per ogni stream trasmesso.
- Il *Player* può essere creato attraverso il metodo *createPlayer* invocato sul *Manager* passando come parametro o il *MediaLocator* oppure il *DataSource*. Se si usa il *MediaLocator* per costruire il Player, è possibile presentare soltanto il primo stream RTP della sessione.
- Se si vuole presentare più stream RTP nella sessione, si deve usare l'*RTPManager* direttamente e costruire un player per ogni *ReceiveStream*.



26

Ricezione e presentazione di stream RTP

- Per ricevere e riprodurre ogni stream ricevuto in una sessione con l'*RTPManager* bisogna seguire i seguenti passi:
- Inizializzare la sessione RTP:
  - Creare un SessionManager. Per esempio, costruire un'istanza della RTPManager (una implementazione di SessionManager):

    RTPManager mgrs= RTPManager.newInstance();
  - Registrarsi come listener:

    mgrs.addSessionListener(this);

    mgrs.addReceiveStreamListener(this);
  - Inizializzare la sessione:

    mgrs.initialize(localAddress);
  - Far partire la sessione:

    mgrs.addTarget(destAddress);
- Nel metodo *update* di *ReceiveStreamListener* gestire l'evento NewReceiveStremEvent che indica la ricezione di un nuovo stream

27

Ricezione e presentazione di stream RTP

- Quando un NewReceiveSteamEvent arriva, si recupera lo stream arrivato col metodo getReceiveStream
- Si ottiene il DataSource dallo stream ricevuto richiamando getDataSource
- Si passa il DataSource al metodo createPlayer del Manager per costruire un nuovo Player. Deve essere disponibile il plug-in per la decodifica e la spacchettizzazione dei dati formattati dello stream.

28

Ricezione e presentazione di stream RTP

Un esempio di utilizzo dell'RTPManager per la ricezione di stream è proposto nella classe AVReceive2.java. Essa si pone in ascolto su due sessioni con indirizzi IP passati come parametri. Per ognuna delle sessioni si aprono tanti Player quanti sono gli stream trasmessi.

In input sono necessarie le due sessioni date secondo il seguente formato:

java AVReceive2 143.225.229.13/48000 143.225.229.13/48002

29

Riferimenti

- Sun Microsystem - JMF:
  http://java.sun.com/products/java-media/jmf/index.jsp
- Gli esempi trattati e molti altri possono essere trovati all'indirizzo:
  http://java.sun.com\products\java-media\jmf\2.1.1\solutions\index.html
- API specification:
  http://java.sun.com/products/java-media/jmf/reference/api/index.html
- Java Media Framework API Guide:
  http://java.sun.com/products/java-media/jmf/2.1.1/guide/index.html

30