Procesadores (Processors)

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

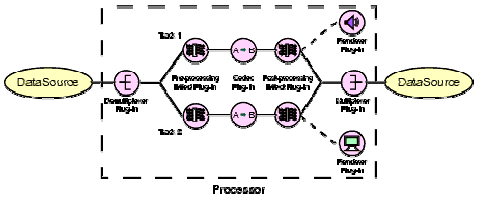
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid

2 Indice

- Repaso
- Controlando un procesador
- Configurando un procesador
- Estableciendo las opciones de procesado de cada pista.
- Especificando el formato de salida del procesador

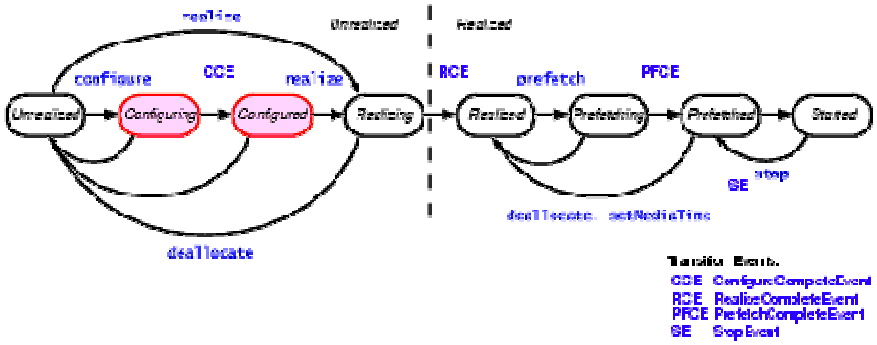
3 Recordemos

- Recordemos que un Processor es un Player programable que permite controlar la decodificación y presentación de los medios.
- Un Processor puede usarse también como capturador para permitir la codificación y multiplexación de los medios capturados antes de su almacenamiento o difusión.
- Un Processor usará diferentes plug-ins instalados en el sistema para llevar a cabo su misión. Mediante la creación de plug-ins podemos extender la JMF.
- Existen cinco tipos de plug-ins:
  - Demultiplexores
  - Efectos
  - Codificadores
  - Renders o presentadores
  - Multiplexores.



4 Recordemos - Los estados de un procesador

- Añade un par de estados previos a los estados del Player:
  - Configuring – tras la llamada al método configure(). Se usa para abrir el DataSource y obtener información del mismo.
  - Configured – al finalizar la configuración. Este estado se puede usar para obtener los TrackControls (con el método getTrackControls()) que sirven para cambiar los plug-ins que se usarán para cada uno de los tracks.



5 Controlando en procesado del Processador

- Tenemos 5 alternativas:
  - Usar un ProcessorModel para construir un Processor que tiene unas características de entrada y salida dadas.
  - Usar el método TrackControl.setFormat para especificar el formato y conversiones en cada track.
  - Usar el método Processor.setContentDescriptor para especificar el formato de datos multiplexados de la salida del Processor.
  - Usar el método TrackControl.setCodecChain para seleccionar el plug-in de Effect o Codec a utilizar en un track particular del Processor.
  - Usar el método TrackControl.setRenderer para seleccionar un Render plug-in a usar en un track particular del Processor.

6 Configurando un Processor.

- Un Processor ofrece el método configure que mueve al procesador al estado de *Configuring* (desde *Unrealized*)
- En el estado de *Configuring*, un Processor recopila la información que necesita para construir los objetos TrackControl para cada pista (track). Cuando acaba se mueve al estado *Configured* y lanza un ConfigureCompleteEvent.
- Una vez que el Processor está en estado *Configured* se pueden establecer el formato de salida y las opciones de control de cada pista (TrackControl options). Una vez especificadas estas opciones se llama al método realize para mover al Processor al estado *Realizing* y seguir con el proceso de configuración al igual que se hacía con los reproductores.
- Una vez que el Processor está en estado *Realized* no se garantiza que funcionen los intentos de modificar las opciones de procesamiento. En muchos casos, esto producirá que se lance una FormatChangeException.

7 Seleccionando las opciones de procesamiento de un track

- Para seleccionar qué plug-ins utilizar dentro de un procesador para el procesamiento de una determinada pista se siguen los siguientes pasos:
  - Invocar el método PlugInManager.getPlugInList para determinar qué plug-ins están disponibles. El PlugInManager devuelve una lista de plug-ins que soportan los formatos de entrada y salida (input-output) especificados.
  - Invocar el método getTrackControls en el Processor para obtener un TrackControl para cada pista (track) en el flujo de medios. El procesador (Processor) debe estar en el estado de *Configured* antes de llamar al método getTrackControls.
  - Invocar el método TrackControl.setCodecChain o setRenderer para especificar los plug-ins que quieres usar para la pista.
- El método setCodecChain permite especificar la cadena a usar tanto de codecs como de procesadores de efectos. El orden de aplicación de estos codecs y p. de efectos se determina por los formatos de entrada y salida de cada uno de ellos.
- El método TrackControl.getControls nos devuelve controles que podemos usar para interactuar desde la aplicación con el procesamiento de la pista (por ejemplo: H263Control, QualityControl, y MPEGAudioControl).

8 Especificar el formato de salida del Processor

- Si se construye el procesador a partir de un ProcessorModel ya se puede especificar los formatos de entrada y salida.
  - **ProcessorModel**(DataSource inputDataSource, Format[] formats, ContentDescriptor outputContentDescriptor)
- Si no se ha especificado el formato de salida a la hora de la creación del procesador se puede invocar el método:
  Processor.setContentDescriptor
- Si queremos saber todos los formatos de salida soportados antes de seleccionar uno de ellos para nuestro procesador podemos invocar el método:
  Processor.getSupportedContentDescriptors

9 Especificando el destino del medio

- Si queremos presentar el medio:
  - Inovocamos getTrackControls en el Processor para obtener el TrackControl para cada track en el flujo de medios. El Processor debe estar en el estado *Configured* antes de invocar getTrackControls.
  - Inovocamos el método TrackControl.setRenderer para especificar el plug-in a utilizar para renderizar el medio del track.
- Para escribir la salida a fichero:
  - Obtener el DataSource de salida del Processor llamando a getDataOutput.
  - Construir un DataSink para escribir a fichero llamando a Manager.createDataSink. Pasarle el DataSource de salida del procesador y un MediaLocator que especifique la localización del fichero en el que escribir.
  - Invocar open en el DataSink para abrir el fichero.
  - Invocar start en el DataSink para empezar a escribir datos.
- El Processor.setContentDescriptor nos sirve para especificar el formato del DataSource de salida del Processor y por lo tanto especificaría el formato de escritura en el fichero.

10 Escribiendo a fichero (ejemplo)

- En la siguiente figura se muestra un código sencillo para ilustrar el proceso de escritura en fichero de la salida del procesador.

```
DataSink sink;
MediaLocator dest = new MediaLocator(file://newfile.wav);
try{
        sink = Manager.createDataSink(p.getDataOutput(), dest);
        sink.open();
        sink.start();
} catch (Exception) {}
```

- Nota: si se especifica un localizador de medios tipo rtp se puede difundir la salida del procesador por la red (lo veremos más adelante).

11 Ejercicio

- Analizar el código:
  - http://java.sun.com/products/java-media/jmf/2.1.1/solutions/Transcode.java
- Y contestar a las siguientes preguntas:
  - ¿para qué sirve el método doIt?
  - ¿Qué hace la aplicación?
  - Buscar documentación sobre wait, notify y synchronized en Java.
  - ¿Cómo se espera a que el procesador esté en estado configured?
  - Comentar qué realiza el método setContentDescriptor y su relación con la salida del procesador
  - Comentar para qué sirve el método setTrackFormats
  - Comentar qué misión tienen los enteros start y end.

12 Repaso (I)

- ¿Cuál es la principal característica que puede distinguir a un Player de un Processor? ¿Cuál de los dos es más potente?
- En un Processor, ¿por qué existe la necesidad dinámica de crear plugins de forma dinámica?
- Diga cuáles son los cinco tipos de plug-ins existentes en un Processor, ¿cuál de ellos nos permite presentar por pantalla un medio?
- Dibuje de forma aproximada el esquema de evento de un Processor, teniendo en cuenta que consta de 8 estados.
- Si se compara con el Player, ¿Qué dos estados nuevos incluye el Processor?
- ¿Qué cinco mecanismos podemos utilizar para controlar un Processor?
- ¿Para qué sirve el método **`configure()`** aplicado sobre un Processor?
- ¿Cómo se puede hacer que el Processor avance hasta el estado de *Realized*?
- Explique que tres pasos hay que dar para seleccionan modificar una determinada pista de audio.

13 Repaso (II)

- Explique para qué sirve el método **`setContentDescriptor`** del objeto Processor
- Explique cómo podemos hacer para que el Processor renderice los datos.
- Idem pero para enviar los datos a un fichero.
- ¿Es necesario llamar al método **`start`**() sobre el Datasink para poder guardar los datos en fichero? Justifique su respuesta

14 Autoría

- Mario Muñoz Organero
- Pablo Basanta Val

  \+ Cuestiones de repaso