



Curso Multimedia Home Platform 1.1.2

Persistent Storage

Accediendo al “disco” del deco.

Curso Multimedia Home Platform 1.1.2

Copyright 2008 © Enrique Pérez Gil

Licensed under the ***Creative Commons Attribution-Non-Commercial-No Derivative Works 3.0 Unported License***. You may not use this file except in compliance with the License. You may obtain a copy of the License at:

<http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode>

This is a human-readable summary of the License applied:

(<http://creativecommons.org/licenses/by-nc-nd/3.0/>)

You are free to Share, to copy, distribute and transmit the work **Under the following conditions:**

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Noncommercial.** You may not use this work for commercial purposes.
- **No Derivative Works.** You may not alter, transform, or build upon this work.

For any reuse or distribution, you must make clear to others the license terms of this work. Any of the above conditions can be waived if you get permission from the copyright holder. Nothing in this license impairs or restricts the author's moral rights.

Introducción

- ¿ Recordamos la propiedad aquella a la que sólo las signed apps tenían acceso ?
dvb.persistent.root
- Pues bien, **sólo las signed** podrán usar un pequeño “**disco duro**” en el STB, cuya estructura de directorios cuelga precisamente del punto definido en la propiedad **dvb.persistent.root**.
- **Como siempre sólo podrán hacerlo si lo solicitan en el PRF (Permission Request File):**

```
<!ELEMENT file EMPTY>  
<!ATTLIST file value (true|false) "true">
```

¿ Qué puedo hacer y qué hay ahí ?

- Este repositorio NO es solo mío, es un lugar compartido con otras APPs.
- Tengo permiso de LECTURA en el directorio raíz, o sea, el que me da la property **dvb.persistent.root**
- Tengo permiso de LECTURA y ESCRITURA en el directorio que cuelga del raíz cuyo nombre es el **ORGID** de mi Xlet (**OJO: en Hexadecimal sin ceros delante...¿con 0x o sin 0x ? SIN 0x, p.e. 3a**). Es decir, se asume que mi organización es consciente de esto y por lo tanto todas sus apps se portarán “bien” manejando los ficheros dentro de ese directorio. **OJO: NO se da acceso a los subdirectorios!!**
- Tengo permiso de LECTURA y ESCRITURA en el directorio que cuelga del ORGID y cuyo nombre es **APPID** (**OJO: en Hexadecimal sin ceros delante**)
- Sólo se puede escribir en un mismo fichero por parte de un solo FileOutputStream.

¿ Hasta cuando duran los ficheros que guarde ahí ?

- Pues lo harán hasta que se dé una de las siguientes situaciones:
 - Si no hay suficiente espacio para una app que está corriendo.
 - Si cuando entra mi APP hay otro directorio que se llama igual que mi APPID cuyo propietario era otra app entonces el contenido del anterior se borra por completo y mi app pasa a ser su dueño.
 - Si el fichero almacenado expira (después vemos qué significa eso)
 - El fichero es eliminado por la aplicación propietaria o por una con permisos (credenciales)
 - El fichero es eliminado como consecuencia de una petición del usuario.
 - El % usado por las aplicaciones MHP del persistent storage (**ojo: no solo lo usa MHP**) excede el 75%. (ved MHP Specs G.7. At least 131 072 bytes)

Un fichero especial para cada ORGID

- Un Terminal MHP no borrará un fichero que se llame **prefs.bin**, que se encuentre bajo el directorio ORGID y cuyo tamaño no sea superior a 2Kbytes, salvo que el persistent storage esté lleno y sólo contenga este tipo de ficheros.
- Esto permite a las organizaciones disponer de un “fichero de propiedades”.

Y ahora el API:

- **org.dvb.io.persistent, Anexo K**
- **java.io.**
- En primer lugar llama la atención que **no** hay que usar DSMCCObject. Podemos usar el API de siempre **java.io**. (en la doc habla de javax.microedition.io pero francamente no sé porqué, quizá sea un bug de la doc).
- En el API de **org.dvb.io.persistent** hay únicamente **2** clases:
 - org.dvb.io.persistent.FileAccessPermissions**
 - org.dvb.io.persistent.FileAttributes**
- Para lo que vale este API es para establecerles a nuestros ficheros creados en el persistent storage **los permisos y atributos que se pueden definir con estas dos clases.**

org.dvb.io.persistent.FileAccessPermissions

- Leyendo los parámetros del constructor podemos entender perfectamente los permisos que se pueden definir, no obstante a continuación veremos las restricciones que se establecen:

readWorldAccessRight : LECTURA para TODAS las apps

writeWorldAccessRight : ESCRITURA para TODAS las apps

readOrganisationAccessRight : LECTURA para TODAS las apps de la misma ORGID

writeOrganisationAccessRight : ESCRITURA para TODAS las apps de la misma ORGID

readApplicationAccessRight : LECTURA para el OWNER

writeApplicationAccessRight : ESCRITURA para el OWNER

```
public FileAccessPermissions(boolean readWorldAccessRight, boolean writeWorldAccessRight,  
    boolean readOrganisationAccessRight, boolean writeOrganisationAccessRight,  
    boolean readApplicationAccessRight, boolean writeApplicationAccessRight)
```

- El resto del API son los get y un set igual al constructor para modificar los permisos.

org.dvb.io.persistent.FileAccessPermissions

- Restricciones: una APP puede modificar los permisos de un fichero del que es propietario de las formas siguientes:
 - Puede dar readonly, writeonly, o readwrite a todas las aplicaciones con el mismo ORGID
 - Puede dar readonly, writeonly, o readwrite a todas las aplicaciones
 - Es necesario disponer de derecho de escritura en un directorio para añadir o quitar ficheros en el mismo
 - Es necesario disponer de derecho de lectura en un directorio para listar su contenido o leer el contenido de un fichero
 - Para leer el contenido de un fichero o directorio se habrá de tener permiso para leer ese fichero o directorio y todos los directorios en el path hasta la raíz del persistent storage.
- **En definitiva**, es necesario, al margen de que la aplicación propietaria facilite los permisos, tener por sí misma acceso a esas ubicaciones (credenciales)

- Ahora bien, ¿ cómo le establezco los Permisos a mi fichero ? ¿ cómo obtengo los permisos de mi fichero ?
- La siguiente clase del API es **FileAttributes**; pues bien, el conjunto de permisos es un “atributo” más y este se establecerá a través de esta nueva clase.

Leyendo y Estableciendo Permisos

- En primer lugar quiero conocer los atributos de mi fichero:

```
public static FileAttributes getFileAttributes(File f) throws IOException
```

- **OJO:** me devuelve UNA COPIA. No será suficiente con modificar el objeto FileAttributes devuelto para modificar los atributos de mi fichero.

- A partir del FileAttributes obtenido puedo saber qué permisos tiene.

```
public FileAccessPermissions FileAttributes.getPermissions()
```

- **OJO:** me devuelve UNA COPIA. No será suficiente con modificar el objeto FileAccessPermissions devuelto para modificar los permisos de mi fichero (sería una copia de una copia).

- Después de cambiar permisos le establezco al FileAttributes el modificado.

```
public void setPermissions(FileAccessPermissions p)
```

- Y finalmente asocio el FileAttributes al File!!!

```
public static void setFileAttributes(FileAttributes p, File f)
```

org.dvb.io.persistent.FileAttributes. Resto del API

- public void **setExpirationDate**(Date d)
Establece la fecha hasta la cual queremos mantenerlo. Después el sistema puede eliminarlo.
- public Date **getExpirationDate**()
- public void **setPriority**(int priority)
 - Se establece la prioridad. A mayor prioridad menos probabilidades de ser eliminado por el sistema

PRIORITY_LOW

PRIORITY_MEDIUM

PRIORITY_HIGH
- public int **getPriority**()

- Por defecto los atributos de un fichero son:
 - low priority,
 - owner read/write only permissions y
 - null expiration date.
- Por cierto, mensajes del DECO arrancando:

....

EventManager: Event Manager created (active=false).

0 = set_attribute_emulation(/flashusr/persistent,1)

2600 = set_quota(/flashusr/persistent,2097152)

KickStart: persistent storage quota: 2097152

KickStart: persistent storage size set to: 2097152

....

Ejercicios Bloque PSTO-1

ISO/IEC 13818-1	Part 1. Elementary Streams transport definition
ISO/IEC 13818-6	Part 6. Extensions for DSM-CC. Digital Storage Media Command and Control
ETSI EN 300 468	Digital Video Broadcasting (DVB);Specification for Service Information (SI) in DVB systems
ETSI EN 301 192	DVB specification for data broadcasting
ETSI TR 101 202	Implementation Guidelines for Data broadcasting
ETSI TR 101 162	Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems
ETSI TR 102 154	Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in Contribution and Primary Dist
ETSI TR 101 211	Guidelines on implementation and usage of Service Information (SI)
ETSI TR 101 200	Digital Video Broadcasting (DVB); A guideline for the use of DVB specifications and standards
DAVIC	Digital Audio Visual Council. davic 1.4.1
HAVI	Specification of the Home Audio/Video Interoperability (HAVi) Architecture
Interactivetvweb	http://www.interactivetvweb.org/
Wikipedia DSMCC	http://en.wikipedia.org/wiki/DSM-CC
MHP 1.1.2	Multimedia Home Platform, A068r1 & tam668r23_11xdraft_20061115
MHP 1.1.3	Multimedia Home Platform, A068r3
CDC 1.1	Connected Device Configuration (CDC) 1.1 (JSR=218).
PBP 1.1	Personal Basis Profile 1.1 (JSR 217)
MHP.org	www.mhp.org
INTRO MHP 1.1.3	tam1032r1-mhp-iptv-presentation