

Présentation de DBus

- 1) Qu'est ce que DBus ?
- 2) Les différents composants de DBus
- 3) Les utilisations actuelles et futures
- 4) Exemples d'utilisations de DBus :
 - Communication avec les applications
 - Utilisation d'un binding Java pour implémenter un HelloWorld
- 5) Conclusion

1) Qu'est ce que Dbus ?

DBus est un système de communication entre processus. C'est une interface qui permet :

- aux applications en cours d'exécution de communiquer en s'échangeant des messages
- au bureau de communiquer avec ces applications
- au bureau de communiquer avec le système d'exploitation

2) Les différents composants

DBus est formé de plusieurs composants :

- Une bibliothèque permettant à deux applications de communiquer : *libdbus*
- Un démon, construit à partir de la bibliothèque *libdbus*, sur lequel les applications peuvent se connecter
- Des bibliothèques d'interconnexions pour de nombreux langages (Python, Java, Qt, C#, Glib...).

Autres implémentations non-officielles pour Ruby, Java et C#

3) Les utilisations

Nombreux outils existants (Bonobo, DCop, Corba, Dcom...).

Actuellement présent sur différents bureaux (Gnome et KDE notamment)

Exemple sur KDE :

```
marco@bing:~$ ps -e | grep dbus
4564 ?          00:00:00 dbus-daemon
6172 ?          00:00:00 dbus-daemon
6173 ?          00:00:00 dbus-launch
marco@bing:~$
```

DBus a été créé en tenant compte de l'expérience acquise avec Corba et DCop et sera normalement utilisé seul sur KDE 4.

4) Exemple 1 (1)

Communication avec une application en cours d'exécution.

Dans cet exemple, nous allons communiquer avec le lecteur multimédia Rhythmbox du projet Gnome.

On lance donc l'application :

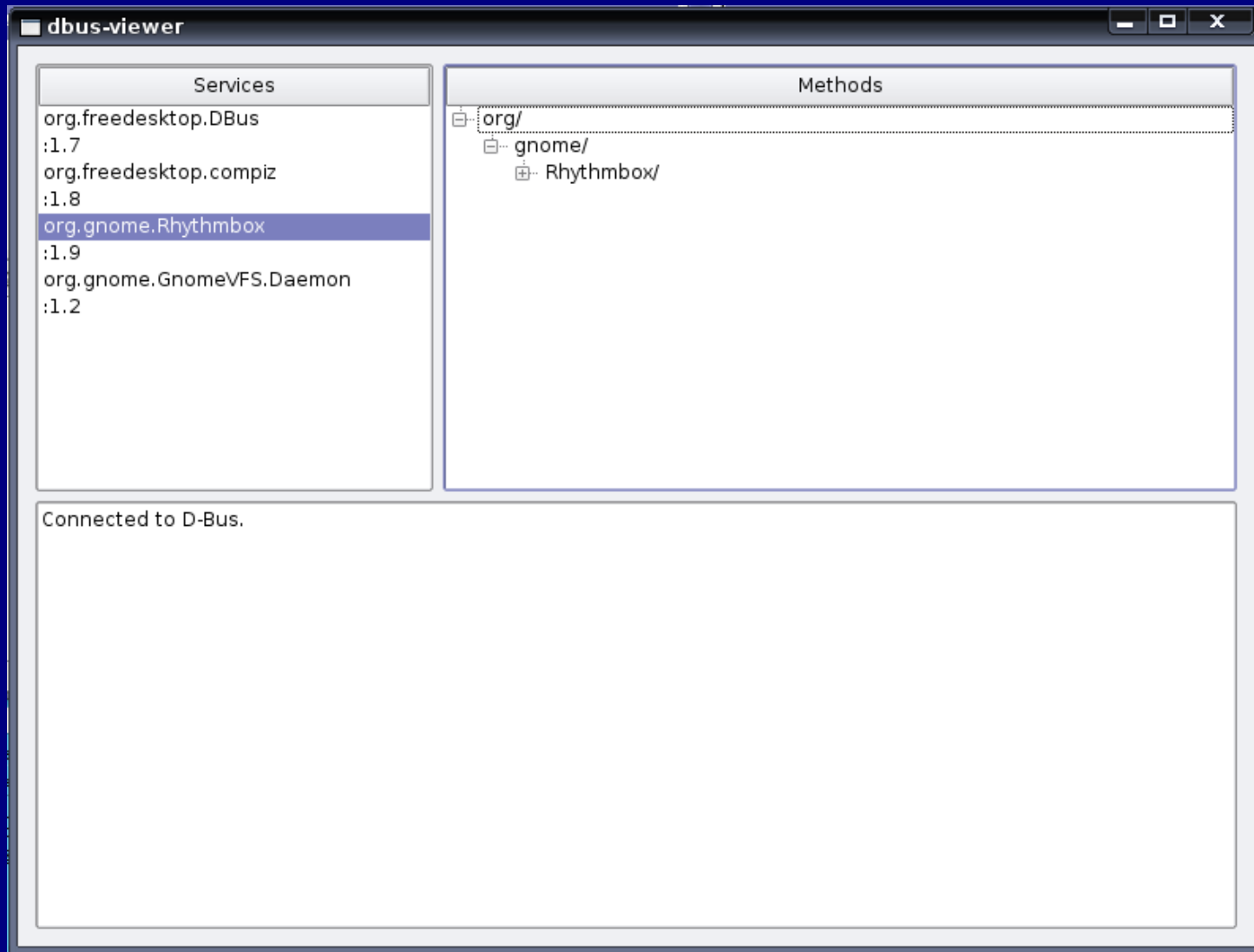
```
rhythmbox &
```

La commande suivante permet d'ouvrir une application graphique nous permettant de visualiser les applications écoutant sur un bus DBus :

```
dbus-viewer &
```

4) Exemple 1 (2)

On remarque donc que l'on peut communiquer avec Rhythmbox :



4) Exemple 1 (3)

Pour connaître les différentes méthodes de Rhythmbox que l'on peut appeler, on utilise le mécanisme d'introspection offert par DBus:

```
dbus-send --session --type=method_call --print-reply --  
dest=org.gnome.Rhythmbox /org/gnome/Rhythmbox/Player  
org.freedesktop.DBus.Introspectable.Introspect
```

Ce mécanisme nous retourne le DTD suivant :

```
string "<!DOCTYPE node PUBLIC "-//freedesktop//DTD D-BUS Object Introspection 1.0//EN" "http://www.freedesktop.org/standards/dbus/1.0/introspect.dtd">
```

```
<node>
```

```
<interface name="org.freedesktop.DBus.Introspectable">
```

```
<method name="Introspect">
```

```
<arg name="data" direction="out" type="s"/>
```

```
</method>
```

```
</interface>
```

```
...
```

```
<interface name="org.freedesktop.DBus.Properties">

    <method name="Get">

        <arg name="interface" direction="in" type="s"/>

        <arg name="propname" direction="in" type="s"/>

        <arg name="value" direction="out" type="v"/>

    </method>

    <method name="Set">

        <arg name="interface" direction="in" type="s"/>

        <arg name="propname" direction="in" type="s"/>

        <arg name="value" direction="in" type="v"/>

    </method>

</interface>

<interface name="org.gnome.Rhythmbox.Player">

    <method name="getMute">

        <arg name="mute" type="b" direction="out"/>

    </method>

    <method name="setMute">

        <arg name="mute" type="b" direction="in"/>

    </method>
```

```
<method name="setVolumeRelative">

    <arg name="volume" type="d" direction="in"/>

</method>

<method name="setVolume">

    <arg name="volume" type="d" direction="in"/>

</method>

<method name="getVolume">

    <arg name="volume" type="d" direction="out"/>

</method>

<method name="setElapsed">

    <arg name="elapsed" type="u" direction="in"/>

</method>

<method name="getElapsed">

    <arg name="elapsed" type="u" direction="out"/>

</method>

<method name="getPlayingUri">

    <arg name="uri" type="s" direction="out"/>

</method>
```

```
<method name="getPlaying">

    <arg name="playing" type="b" direction="out"/>

</method>

<method name="next">

</method>

<method name="previous">

</method>

<method name="playPause">

    <arg name="arg0" type="b" direction="in"/>

</method>

<signal name="playingSongPropertyChanged">

    <arg type="s"/>

    <arg type="s"/>

    <arg type="v"/>

    <arg type="v"/>

</signal>

...
```


4) Exemple 1 (4)

On va maintenant utiliser Dbus pour interagir avec l'application, en utilisant les méthodes retournées par le DTD.

Par exemple, on charge une liste de lecture dans Rhythmbox, puis on lance la lecture avec la commande :

```
dbus-send --session --type=method_call --print-reply --  
dest=org.gnome.Rhythmbox /org/gnome/Rhythmbox/Player  
org.gnome.Rhythmbox.Player.playPause boolean:true
```

Pour lire le fichier suivant dans la liste de lecture, on appellera la méthode next :

```
dbus-send --session --type=method_call --print-reply --  
dest=org.gnome.Rhythmbox /org/gnome/Rhythmbox/Player  
org.gnome.Rhythmbox.Player.next
```

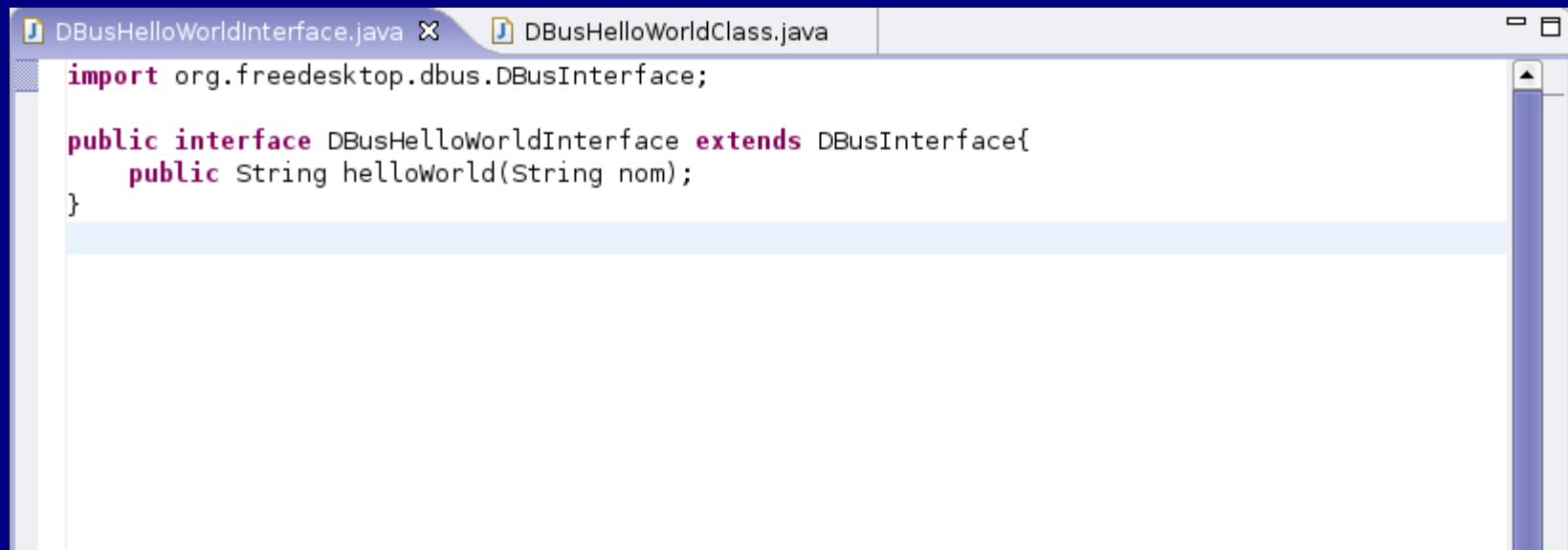
4) Exemple 2 (1)

Utilisation d'un binding pour communiquer avec une application Java.

L'idée est de créer une application qui va tourner en tâche de fond (un serveur en quelque sorte), et qui va proposer un certain nombre de méthodes qui vont être appelables depuis l'extérieur.

Dans cet exemple il s'agit de faire un Hello World, callable depuis un terminal.

On définit tout simplement une interface, puis une classe qui implémente cette interface et qui définit la méthode que l'on souhaite appeler :

A screenshot of a Java IDE window with two tabs: 'DBusHelloWorldInterface.java' and 'DBusHelloWorldClass.java'. The 'DBusHelloWorldInterface.java' tab is active, showing the following code:

```
import org.freedesktop.dbus.DBusInterface;

public interface DBusHelloWorldInterface extends DBusInterface{
    public String helloWorld(String nom);
}
```

4) Exemple 2 (2)

Voici le code de la classe :

```
DBusHelloWorldInterface.java  DBusHelloWorldClass.java ✕

import org.freedesktop.dbus.DBusConnection;
import org.freedesktop.dbus.exceptions.DBusException;

public class DBusHelloWorldClass implements DBusHelloWorldInterface
{
    private DBusConnection dbusConnection;

    public boolean isRemote()
    {
        return false;
    }

    private boolean stop=false;
    public String helloWorld(String name)
    {
        stop=true;
        return "Hello World : "+name;
    }

    public void start()
    {
        try
        {
            dbusConnection = DBusConnection.getConnection(DBusConnection.SESSION);
            dbusConnection.requestBusName("mon.premier.bus");
            dbusConnection.exportObject("/Main", this);
            while (!stop)
            {
                try
                {
                    Thread.sleep(1000);
                } catch (Exception e) {}
            }
            dbusConnection.disconnect();
        } catch (DBusException e)
        {
            e.printStackTrace();
        }
    }

    public static void main(String[] args)
    {
        new DBusHelloWorldClass().start();
    }
}
```

4) Exemple 2 (3)

Il suffit maintenant de compiler les composants, puis de lancer notre serveur :

```
javac *.java
```

```
java DbusHelloWorldClass &
```

Ceci fait, on utilise la GUI Dbus-Viewer comme dans l'exemple 1 pour utiliser l'introspection et connaître la méthode à utiliser, puis on envoie un message depuis un terminal :

```
dbus-send --print-reply --dest='mon.premier.bus' /Main  
mon.premier.bus.helloWorld string:'Marco'
```

Le serveur nous retourne alors la chaîne suivante, avant de s'arrêter :

```
string "Hello World : Marco"
```

5) Conclusion

- Dbus va bientôt s'imposer comme interface de communication inter-processus. Son utilisation se révèle assez facile, d'autant plus que de nombreux bindings existent et facilitent la tâche aux développeurs.
- La plupart des logiciels existants utilisent une autre technologie, seront-ils adaptés pour utiliser Dbus, ou faudra-t'il utiliser des « ponts » pour connecter toutes les applications ?

Sources

- Le site officiel de DBus : <http://www.freedesktop.org/wiki/Software/dbus>
- Le tutorial officiel présentant les différentes fonctionnalités offertes par DBus : <http://dbus.freedesktop.org/doc/dbus-tutorial.html>
- Une mini-introduction à DBus : <http://dedgui.free.fr>
- Une introduction au binding Java : <http://artisan.karma-lab.net/mise-en-oeuvre-de-dbus-sous-java>