

```
static void
properties(GObjectClass
            *gobject_class)
{
    GObjectSpec *pspec;

    gobject_class->
        attribute */
        CODE,
        "code.",
        "code",
        0,
        74,
        0,
        7,
        |
        |
        ;
}
```

itoral@igalia.com





Slides or source code?

Get the sample source here:
<http://blogs.igalia.com/itoral>

About myself

Member of Igalia since 2003.

Became a partner of the company in 2005.

Member of the build-brigade.

<http://live.gnome.org/BuildBrigade>

<http://build.gnome.org>

Developer of MAFW.

Also interested in GStreamer.



Index

Quick introduction to MAFW.

Writing source plugins.

Writing renderer plugins.

Testing your plugins.



Quick Introduction To MAFW

Quick Introduction To MAFW

Media Application Framework for Maemo.

High level APIs, close to application use cases.

Wider approach than traditional frameworks:

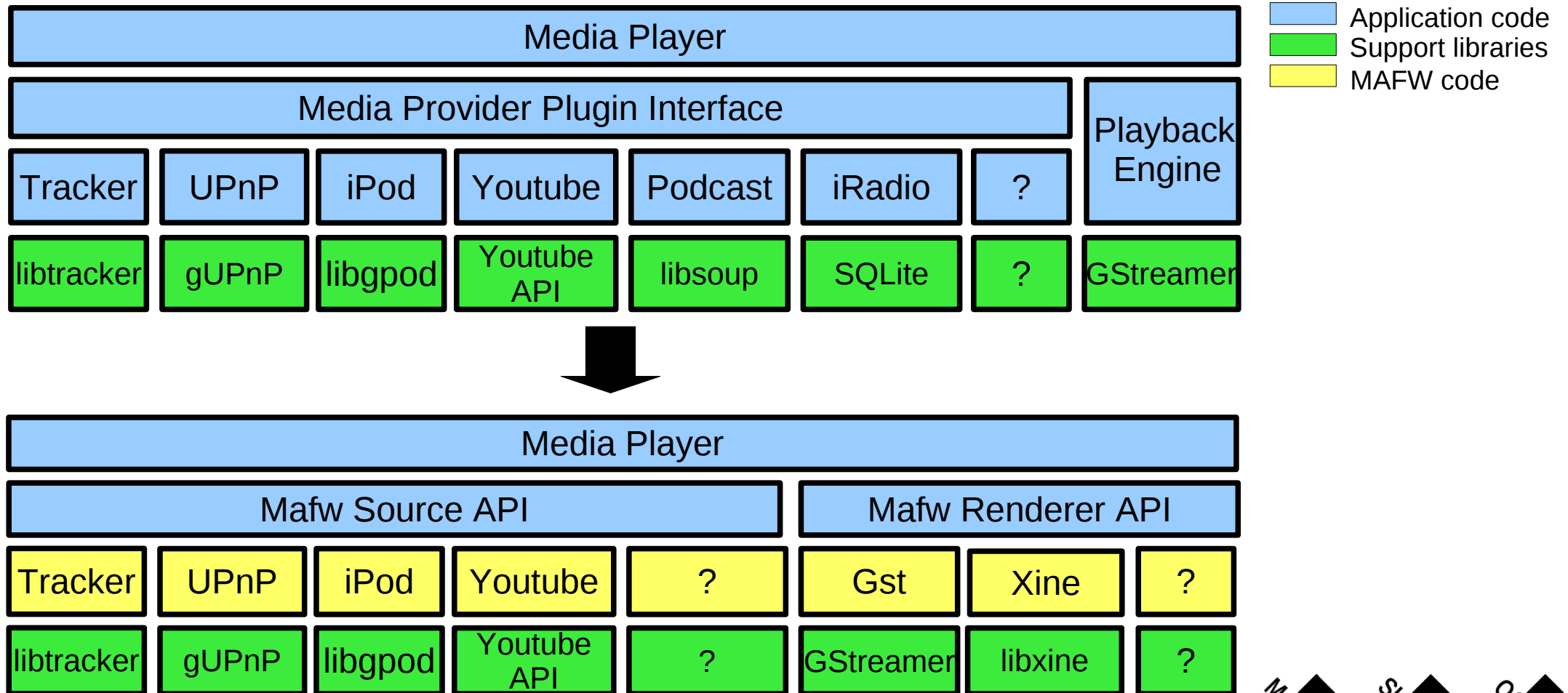
- Media discovery.
- Playlist management.

Plugin based architecture.

Open Source.



Quick Introduction To MAFW



Quick Introduction To MAFW

- Extensions:
 - Sources:
 - Provide access to media content exposing a tree of content that users can browse:
 - Categories: Containers of content, can contain other categories or media items (example: artists, albums, etc).
 - Media: Any resource that can be played.
 - Each item exposed by a source object has a unique identifier (object identifier).
 - <source-uuid>::<resource-identifier>



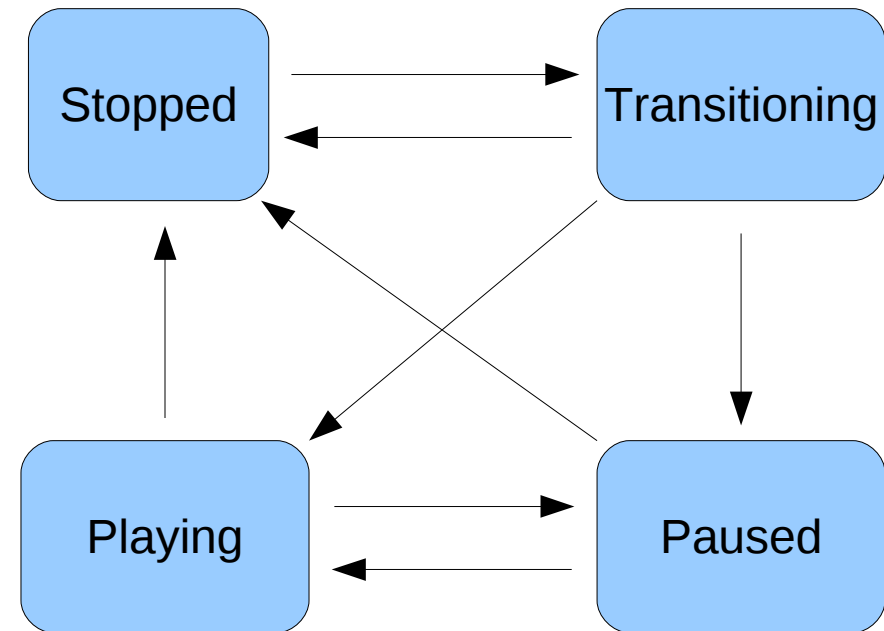
Quick Introduction To MAFW

- Extensions:
 - Renderers:
 - Provide means to play back content exposed by the sources.
 - They implement the playback pipeline.
 - They do not play individual items, but entire playlists.
 - Expose API to control playback:
 - Play, Pause, Resume, Stop, Next, Previous, etc
 - Since playlists in MAFW are a set of object identifiers, the renderers must communicate with the sources to retrieve the URI of the actual items contained in the playlist.



Quick Introduction To MAFW

- Extensions:
 - Renderers:
 - State machine.



Writing Source Plugins

Writing Source Plugins

- Inherit from MafwSource.
- APIs to implement (minimum set):
 - mafw_source_browse
 - mafw_source_get_metadata(s)
- Signals to emit:
 - Container-changed, metadata-changed, updating

Writing Source Plugins

- Important metadata to support:
 - MAFW_METADATA_KEY_{URI, MIME, TITLE}
- Other considerations:
 - Do not block, use the idle loop.
 - MAFW_SOURCE_ALL_KEYS

Writing Renderer Plugins

Writing Renderer Plugins

- Extend from MafwRenderer.
- APIs to implement (there are some more):
 - mafw_renderer_{assign_playlist, play, pause, resume, stop}
- Signals to emit:
 - playlist-changed, media-changed, state-changed, buffering-info, metadata-changed

Writing Renderer Plugins

- Considerations:
 - Renderers implement the playlist logic:
 - Move to next on EOS / error / etc
 - Next, Prev, GotoIndex, etc
 - MafwPlaylist::contents-changed
 - Increment and decrement playlist use count.

Testing Your Plugins

Testing Your Plugins

- Start maemo environment:
 - `af-sb-init start`
 - `run-standalone.sh mafw-playlist-daemon`
 - `run-standalone.sh mafw-dbus-wrapper <plugin.so>`
- Mafw-test-gui
 - Sandbox application, works with any plugin.



Thanks!

