TimeSide, an open web audio processing framework

Guillaume Pellerin[1]

[1]Parisson, Paris, France

SoundSoftware 2014
3rd Workshop on Software and Data for Audio and Music Research
Queen Mary University of London - 08/07/2014

Table of contents

Main goals

- Archive, preserve and scale big music data and related metadata
- Play audio and read metadata synchronously
- Index and share music data through a collaborative web app
- Link music data to various ontologies and external services
- Manage workflow rules (access, copyrights) easily through time
- On demand audio processing through a modular architecture

History of the project

- 2006: definition of the goals (open source web audio collaborative platform)
- 2007: first partner: french Research Center of Ethnomusicology (CREM)
- 2007 - 2009: technical specifications, definition of the DB migrator
- 2008: prototype development
- 2008 - 2010: workflow and format specifications
- 2011: development, final migration and release of Telemeta 1.0 to the CREM for production : http://archives.crem-cnrs.fr
- 2011 - 2014: collaborative indexing, more development, massive data imports...

Technologies

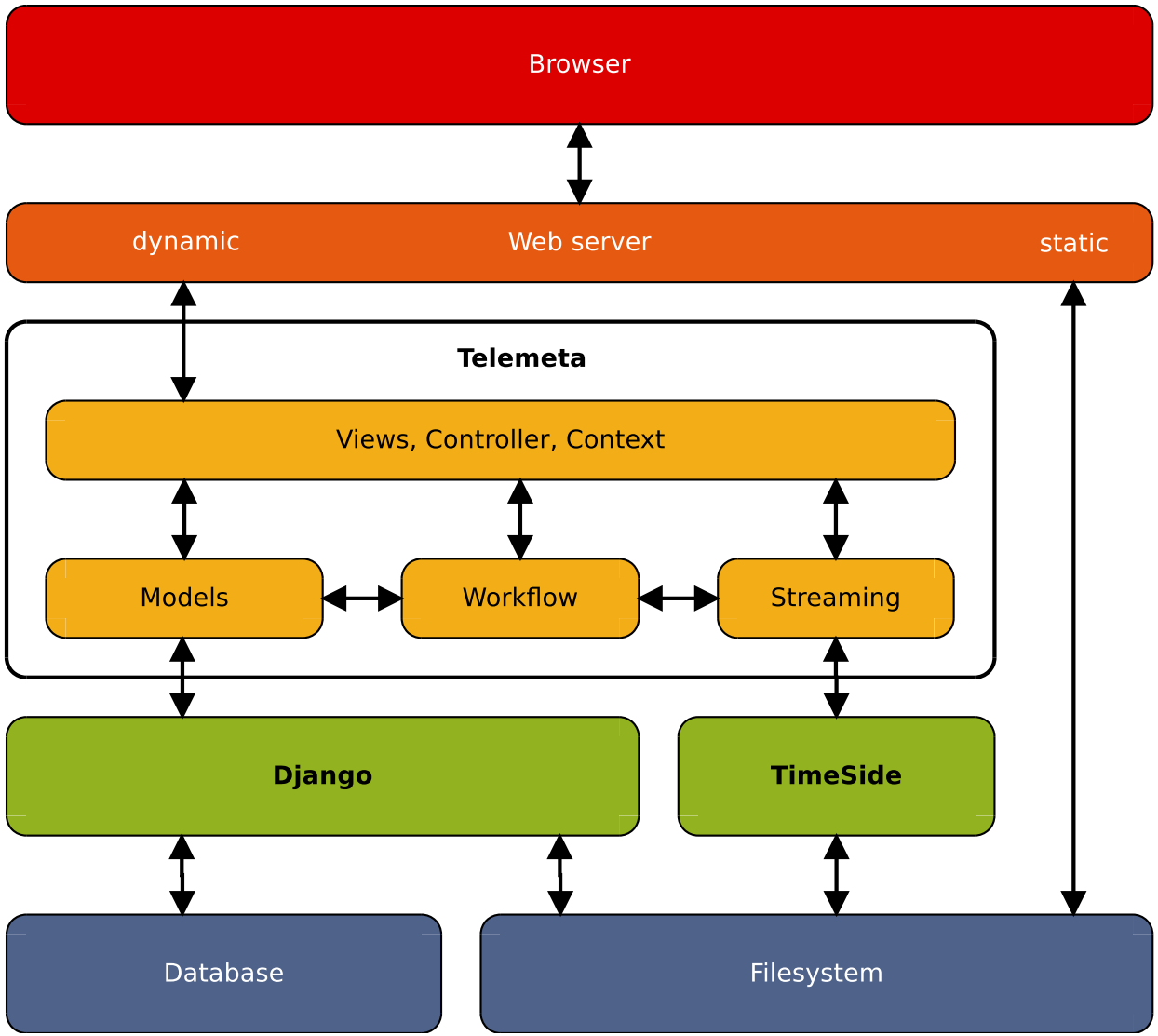
100% 0pen Source!

- Python : smart object oriented language
- Django : high-level web MVC framework
- TimeSide : open web audio processing framework
- GStreamer : open source multimedia framework
- MySQL, PostgreSQL, others : relational databases
- GNU / Linux : applications, libraries and kernel

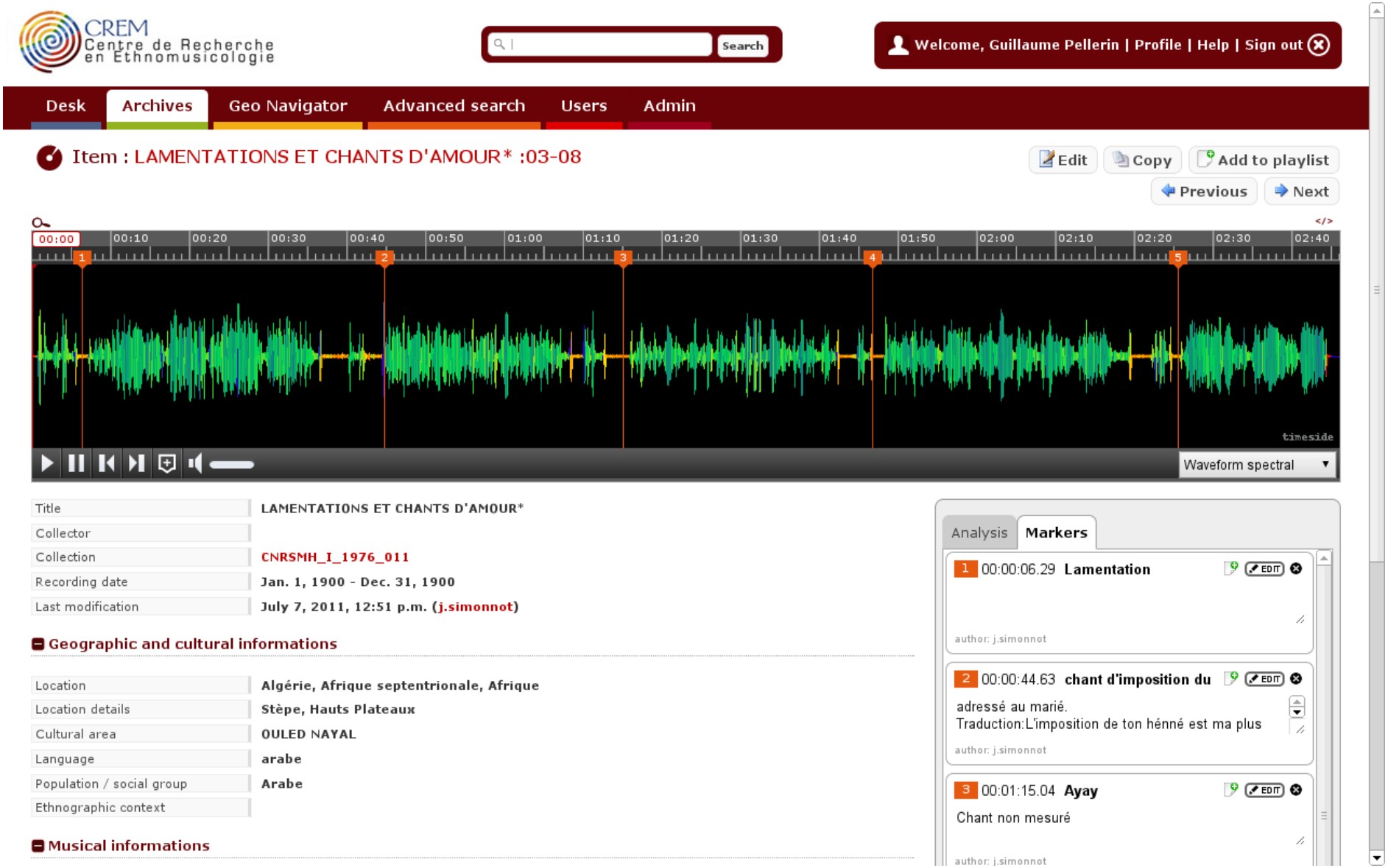
Key features

- Pure HTML5 web user interface including dynamical forms and smart workflows
- On the fly audio analyzing, transcoding and metadata embedding in various formats
- Social editing with semantic ontologies, smart workflows, realtime tools, human or automatic annotations and segmentations
- User management with individual desk, playlists, profiles and access rights
- High level search engine (geolocation, instruments, ethnic groups, etc...)
- Data providers : DublinCore, OAI-PMH, RSS, XML, JSON and other
- Multi-language support (now english and french)

Architecture



Telemeta Web User Interface



http://archives.crem-cnrs.fr/archives/items/CNRSMH_I_1976_011_003_08/

TimeSide : open web audio processing framework

Server side - TimeSide Engine

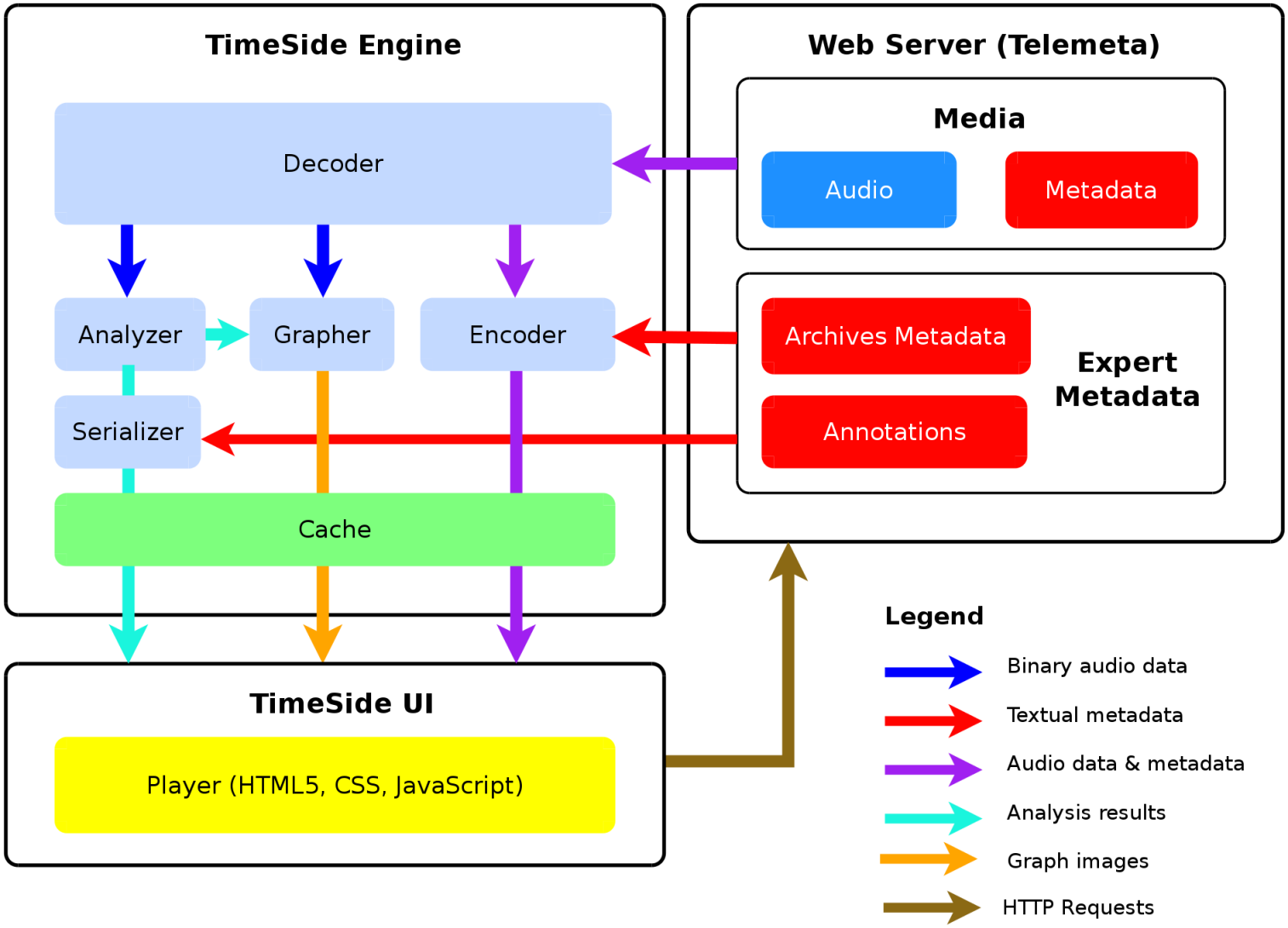
- Do asynchronous and fast audio processing with Python,
- Decode audio frames from ANY format into numpy arrays,
- Analyze audio content with state-of-the-art audio feature extraction libraries (Aubio, Yaafe, Vamp (experimental),
- Organize, serialize and save analysis metadata through various formats,
- Draw various fancy waveforms, spectrograms and other cool graphers,
- Transcode audio data in var_ious media formats and stream them through web apps,

Client side - TimeSide UI

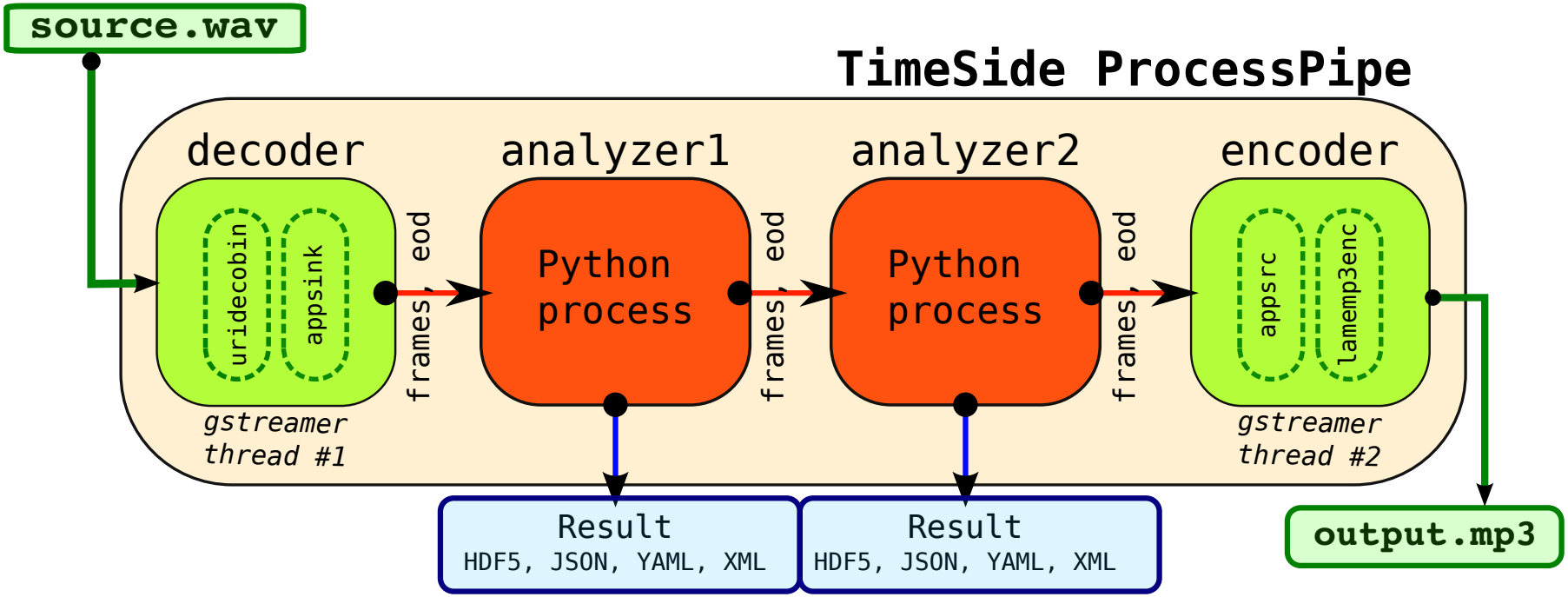
- Playback and interact on demand through a smart high-level HTML5 extensible player,
- Index, tag and organize semantic metadata

  (see Telemeta which embeds TimeSide).

TimeSide - Architecture



TimeSide Engine




```
pipe = (decoder | analyzer1 | analyzer2 | encoder)
```

Process Pipe

- On-the-fly audio processing by simultaneous processors (decoder, encoders, analyzers, graphers)
- Use of *Gstreamer* for audio decoding and encoding

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

Analyzer Result

Result types: *time mode* x *data mode*

- Data modes:
  - Label
  - Value
- Time modes:
  - Global
  - Event
  - Segment
  - Framewise

- ID Metadata
- Audio Metadata
- Data object
  - Label
  - Label Metadata (label, label_id, ...)
  - Value
  - Time
  - Duration
  - Frame Metadata (sample rate, blocksize, stepsize)
- Parameters

The DIADEMS project

- DIADEMS : Description, Indexation, Access to Sound and Ethnomusicological Documents
- Granted by ANR : french national research agency (ANR-12-CORD-0022)
- 3 years, 8 partners, 850 k€
- Apply and test MIR algorithms on large scale ethnomusicological data
- Define some high level interfaces to find new ways of explorations in large complex musical corpus
- New modes of collaboration between human science and computer science laboratories and researchers
- Define the vocabulary describing musical events in the usecase of ethnomusicilogy vs. signal processing
- http://www.irit.fr/recherches/SAMOVA/DIADEMS/fr/welcome/
- http://diadems.telemeta.org

DIADEMS - Partners

- Sponsors:
  - CNRS
  - Huma-Num (ex TGE Adonis)
  - ANR
  - CREM
  - UPMC
  - Parisson
- Partners :
  - IRIT (université Paul Sabatier, Toulouse 3)
  - LIMSI (universités Pierre et Marie Curie (UPMC, Paris 6) et Paris-Sud)
  - LAM (institut Jean Le Rond d'Alembert, UPMC)
  - LABRI (université de Bordeaux)
  - CREM (université Paris Ouest Nanterre La Défense)
  - LESC (université Paris Ouest Nanterre La Défense)
  - Museum d'Histoire Naturelle de Paris
  - Musée du Quai Branly

Development

Links

- http://telemeta.org
- https://github.com/yomguy/Telemeta/
- https://github.com/yomguy/TimeSide/

Team

- Guillaume Pellerin
- Thomas Fillon
- Paul Brossier
- Riccardo Zaccarelli
- Maxime Lecoz
- David Doukan

Licence

CeCILL v2.1 (GPL v2 compatible)

Development - Lessons from a 7 year old project

- Simplicity is better than complexity (a Python developer rule)
- Modularity is only accessible with a flexible language
- Models and objects are more important than technologies
- A good workflow is defined by the users themselves through feedback and revisions
- Prototyping is a crucial part of the development process
- A good platform should rely on standards, not on formats
- The Open Source ecosystem gives some tremendous possibilities to scale a platform project

Development - TODO list

TimeSide

- Tiny web server (django)
- Process task manager
- Full HTML5 zooming player (+ annotations, segmentations, etc..)
- Analyzer parameters (+ interface)
- Improve Vamp plugins support (Vamp python host ?)
- Add more automatic segmentation and classification tools to support various semantic ontologies (cf. thesaurus)
- Add more music analysis tools to support Ethnomusicological research
- Add automatic similarity analysis tools (inside a song or between sound items)
- Enhance analysis result displays to send to Telemeta
- https://github.com/yomguy/TimeSide/issues

Development - TODO list

Telemeta

- Update code to support Django new Class based views
- Rewrite geolocation services
- Public and enhanced user playlists
- Smart breadcrumbs
- Better interactions with TimeSide
- Enhance user interface (full HTML 5 + web audio API)
  - For annotations and segmentations in a collaborative manner
  - Provide import capabilities and feedback loop between manual and automatic annotations
  - Fancy displays of automatic analysis results (zoomable + synchronized with audio)
  - Add a User interface to control and tune the analysis parameters
- More documentation
- http://telemeta.org/report/1

The End

Thank you!

Links

- telemeta.org
- @telemeta

Contact

- guillaume@parisson.com
- @yomguy
- github.com/yomguy/
- +GuillaumePellerin
- fr.linkedin.com/in/guillaumepellerin

TimeSide - Github repository

https://github.com/yomguy/TimeSide/

- 3 main branches: master, dev, diadems

Installation

https://github.com/yomguy/TimeSide#install

- Installation des dépendances :

```
$ echo "deb http://debian.parisson.com/debian/ stable main" |
$ sudo tee -a /etc/apt/sources.list
$ echo "deb-src http://debian.parisson.com/debian/ stable main" | sudo tee -a /etc/apt/sources.list
$ sudo apt-get update
$ sudo apt-get install git
$ sudo apt-get build-dep python-timeside
```

- Installation depuis le dépôt *Github* :

```
$ git clone https://github.com/yomguy/TimeSide.git
$ cd TimeSide
$ git checkout dev
$ export PYTHONPATH=$PYTHONPATH:`pwd`
$ python tests/run_all_tests
```