Sustainability of ‘live electronic’ music in the Integra project

James Bullock[1] Henrik Frisk[2] Lamberto Coccioli[1]

[1]Music Technology Department at Birmingham Conservatoire
Birmingham City University

[2]Composition Department at Malmö Academy of Music
Lund University

Melecon 08

Integra

What is Integra?

- “A European Composition and Performance Environment for Sharing Live Music Technologies"
- 3-year project led by Birmingham Conservatoire in the UK
- Attempts to address the problems of persistent storage, portability and standardised intercommunication between systems for electronic music.
- All data relating to a musical work should be stored.
- The data should be transferable to a variety of useful targets.

Integra

What is Integra?

- “A European Composition and Performance Environment for Sharing Live Music Technologies"
- 3-year project led by Birmingham Conservatoire in the UK
- Attempts to address the problems of persistent storage, portability and standardised intercommunication between systems for electronic music.
- All data relating to a musical work should be stored.
- The data should be transferable to a variety of useful targets.

Integra

What is Integra?

- “A European Composition and Performance Environment for Sharing Live Music Technologies"
- 3-year project led by Birmingham Conservatoire in the UK
- Attempts to address the problems of persistent storage, portability and standardised intercommunication between systems for electronic music.
- All data relating to a musical work should be stored.
- The data should be transferable to a variety of useful targets.

Integra

What is Integra?

- “A European Composition and Performance Environment for Sharing Live Music Technologies"
- 3-year project led by Birmingham Conservatoire in the UK
- Attempts to address the problems of persistent storage, portability and standardised intercommunication between systems for electronic music.
- All data relating to a musical work should be stored.
- The data should be transferable to a variety of useful targets.

Integra

What is Integra?

- “A European Composition and Performance Environment for Sharing Live Music Technologies"
- 3-year project led by Birmingham Conservatoire in the UK
- Attempts to address the problems of persistent storage, portability and standardised intercommunication between systems for electronic music.
- All data relating to a musical work should be stored.
- The data should be transferable to a variety of useful targets.

Integra

Core components

- A software library: libIntegra
- A database
- An XML file format: IXD (Integra eXtensible Data)

The Integra module

is the abstract definition of data shared between these resources.

Integra

Core components

- A software library: libIntegra
- A database
- An XML file format: IXD (Integra eXtensible Data)

The Integra module

is the abstract definition of data shared between these resources.

Integra

Core components

- A software library: libIntegra
- A database
- An XML file format: IXD (Integra eXtensible Data)

The Integra module

is the abstract definition of data shared between these resources.

Integra

Core components

- A software library: libIntegra
- A database
- An XML file format: IXD (Integra eXtensible Data)

The Integra module

is the abstract definition of data shared between these resources.

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma http://www.jamoma.org/
  - Jade http://www.electrotap.com/jade
- Emphasis on software independence:
  - Faust http://faust.grame.fr
  - NASPRO http://sourceforge.net/projects/naspro/
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project http://www.casparpreserves.eu/

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

Related work

- Standardized DSP module construction, providing a generalized namespace:
  - Jamoma `http://www.jamoma.org/`
  - Jade `http://www.electrotap.com/jade`
- Emphasis on software independence:
  - Faust `http://faust.grame.fr`
  - NASPRO `http://sourceforge.net/projects/naspro/`
- Music documentation and migration efforts:
  - PD Repertory Project
  - Mustica
  - CASPAR project `http://www.casparpreserves.eu/`

Integra

The Integra module

- An Integra module encapsulates a specific piece of message or signal processing functionality (e.g. a waveform generator or digital filter).
- A module
  - must have an interface definition
  - may have an implementation
  - may be associated with instance data.
- A module may inherit the interface from any other module.
- A module definition can be thought of as an abstract class.

Only the implementation is platform specific.

Integra

The Integra module

- An Integra module encapsulates a specific piece of message or signal processing functionality (e.g. a waveform generator or digital filter).
- A module
  - must have an interface definition
  - may have an implementation
  - may be associated with instance data.
- A module may inherit the interface from any other module.
- A module definition can be thought of as an abstract class.

Only the implementation is platform specific.

Integra

The Integra module

- An Integra module encapsulates a specific piece of message or signal processing functionality (e.g. a waveform generator or digital filter).
- A module
  - must have an interface definition
  - may have an implementation
  - may be associated with instance data.
- A module may inherit the interface from any other module.
- A module definition can be thought of as an abstract class.

Only the implementation is platform specific.

Integra

The Integra module

- An Integra module encapsulates a specific piece of message or signal processing functionality (e.g. a waveform generator or digital filter).
- A module
  - must have an interface definition
  - may have an implementation
  - may be associated with instance data.
- A module may inherit the interface from any other module.
- A module definition can be thought of as an abstract class.

Only the implementation is platform specific.

Integra

The Integra module

- An Integra module encapsulates a specific piece of message or signal processing functionality (e.g. a waveform generator or digital filter).
- A module
  - must have an interface definition
  - may have an implementation
  - may be associated with instance data.
- A module may inherit the interface from any other module.
- A module definition can be thought of as an abstract class.

Only the implementation is platform specific.

Integra

The Integra module

- An Integra module encapsulates a specific piece of message or signal processing functionality (e.g. a waveform generator or digital filter).
- A module
  - must have an interface definition
  - may have an implementation
  - may be associated with instance data.
- A module may inherit the interface from any other module.
- A module definition can be thought of as an abstract class.

Only the implementation is platform specific.

Integra

The Integra module

- An Integra module encapsulates a specific piece of message or signal processing functionality (e.g. a waveform generator or digital filter).
- A module
  - must have an interface definition
  - may have an implementation
  - may be associated with instance data.
- A module may inherit the interface from any other module.
- A module definition can be thought of as an abstract class.

Only the implementation is platform specific.

Integra

The Integra module

- An Integra module encapsulates a specific piece of message or signal processing functionality (e.g. a waveform generator or digital filter).
- A module
  - must have an interface definition
  - may have an implementation
  - may be associated with instance data.
- A module may inherit the interface from any other module.
- A module definition can be thought of as an abstract class.

Only the implementation is platform specific.

Integra

The module definition

Field	Value
Name	Oscillator
Parent	Module
Attributes	freq, phase
Attribute Unit Codes	1, 2
Attribute Minima	0, 0
Attribute Maxima	inf, 6.2831853071795862
Attribute Defaults	440, 0

Integra

The module definition

Field	Value
Name	Oscillator
Parent	Module
Attributes	freq, phase
Attribute Unit Codes	1, 2
Attribute Minima	0, 0
Attribute Maxima	inf, 6.2831853071795862
Attribute Defaults	440, 0

Integra

The module definition

Field	Value
Name	Oscillator
Parent	Module
Attributes	freq, phase
Attribute Unit Codes	1, 2
Attribute Minima	0, 0
Attribute Maxima	inf, 6.2831853071795862
Attribute Defaults	440, 0

Integra

The module definition

Field	Value
Name	Oscillator
Parent	Module
Attributes	freq, phase
Attribute Unit Codes	1, 2
Attribute Minima	0, 0
Attribute Maxima	inf, 6.2831853071795862
Attribute Defaults	440, 0

Integra

The module definition

Field	Value
Name	Oscillator
Parent	Module
Attributes	freq, phase
Attribute Unit Codes	1, 2
Attribute Minima	0, 0
Attribute Maxima	inf, 6.2831853071795862
Attribute Defaults	440, 0

Integra

The module definition

Field	Value
Name	Oscillator
Parent	Module
Attributes	freq, phase
Attribute Unit Codes	1, 2
Attribute Minima	0, 0
Attribute Maxima	inf, 6.2831853071795862
Attribute Defaults	440, 0

Integra

The module definition

Field	Value
Name	Oscillator
Parent	Module
Attributes	freq, phase
Attribute Unit Codes	1, 2
Attribute Minima	0, 0
Attribute Maxima	inf, 6.2831853071795862
Attribute Defaults	440, 0

Integra

The module namespace

- The namespace is derived from the definition.
- As an OSC (Open Sound Control) address space:

OSC address	Purpose
/oscillator/freq <value>	Set the value of the ‘freq’ attribute
/oscillator/phase <value>	Set the value of the ‘phase’ attribute
/module/active <value>	Set whether or not the module is active

Integra

The module namespace

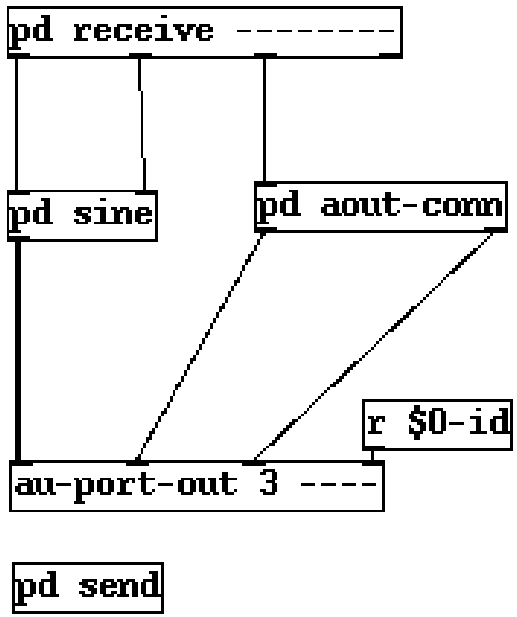
- The namespace is derived from the definition.
- As an OSC (Open Sound Control) address space:

OSC address	**Purpose**
/oscillator/freq <value>	Set the value of the ‘freq’ attribute
/oscillator/phase <value>	Set the value of the ‘phase’ attribute
/module/active <value>	Set whether or not the module is active

Integra

The module implementation

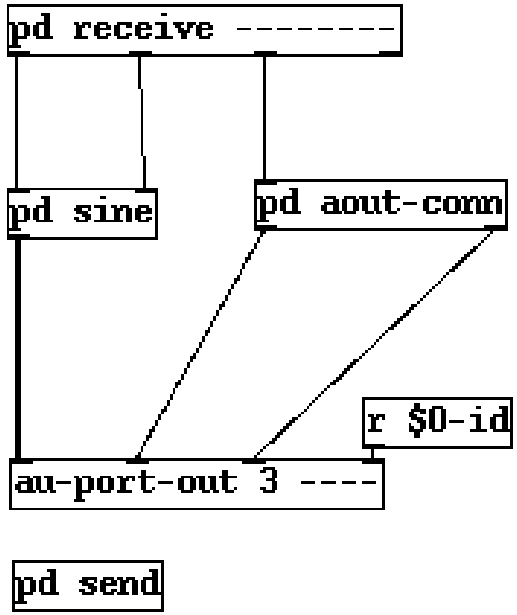
- The only software-specific data stored by Integra.
- Each supported target has its own implementation protocol, *a bridge*.
- Bridges have been developed for Pd and Max/MSP.
- Pd implementation of Sinus module:



Integra

The module implementation

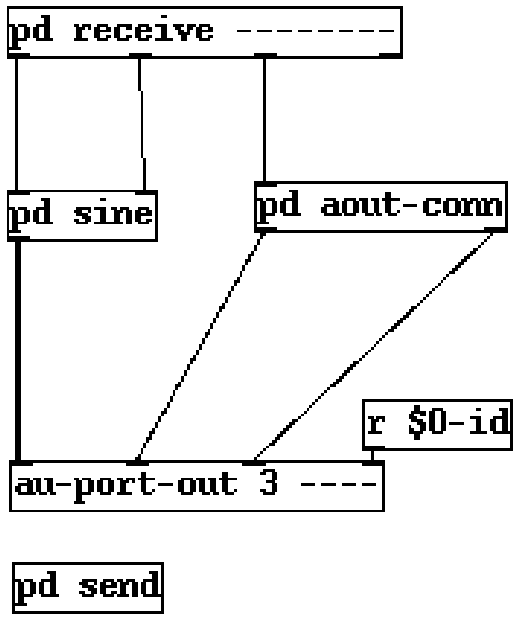
- The only software-specific data stored by Integra.
- Each supported target has its own implementation protocol, *a bridge*.
- Bridges have been developed for Pd and Max/MSP.
- Pd implementation of Sinus module:



Integra

The module implementation

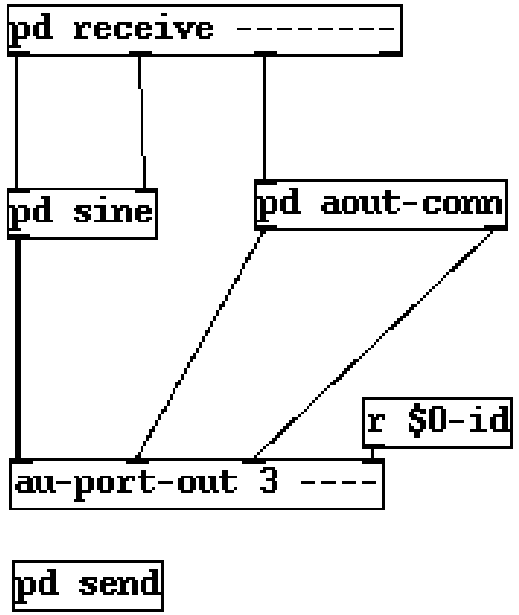
- The only software-specific data stored by Integra.
- Each supported target has its own implementation protocol, *a bridge*.
- Bridges have been developed for Pd and Max/MSP.
- Pd implementation of Sinus module:



Integra

The module implementation

- The only software-specific data stored by Integra.
- Each supported target has its own implementation protocol, *a bridge*.
- Bridges have been developed for Pd and Max/MSP.
- Pd implementation of Sinus module:



Integra

Module instance data

- The run-time state of each module is recorded by the library.
- May be written to XML on demand.
- The run-time state of an instance is stored in its instance table in the database.

Integra

Module instance data

- The run-time state of each module is recorded by the library.
- May be written to XML on demand.
- The run-time state of an instance is stored in its instance table in the database.

Integra

Module instance data

- The run-time state of each module is recorded by the library.
- May be written to XML on demand.
- The run-time state of an instance is stored in its instance table in the database.

Integra

Module collections

- A module collection is one or more interconnected module instances.
- A Collection can
  - contain other collections
  - 'hide' some of its parameters
  - store the current state of each of its instantiated modules
  - be written to XML and stored along with its current state.

Integra

Module collections

- A module collection is one or more interconnected module instances.
- A Collection can
  - contain other collections
  - ‘hide’ some of its parameters
  - store the current state of each of its instantiated modules
  - be written to XML and stored along with its current state.

Integra

Module collections

- A module collection is one or more interconnected module instances.
- A Collection can
  - contain other collections
  - ‘hide’ some of its parameters
  - store the current state of each of its instantiated modules
  - be written to XML and stored along with its current state.

Integra

Module collections

- A module collection is one or more interconnected module instances.
- A Collection can
  - contain other collections
  - ‘hide’ some of its parameters
  - store the current state of each of its instantiated modules
  - be written to XML and stored along with its current state.

Integra

Module collections

- A module collection is one or more interconnected module instances.
- A Collection can
  - contain other collections
  - ‘hide’ some of its parameters
  - store the current state of each of its instantiated modules
  - be written to XML and stored along with its current state.

Integra

Module collections

- A module collection is one or more interconnected module instances.
- A Collection can
  - contain other collections
  - ‘hide’ some of its parameters
  - store the current state of each of its instantiated modules
  - be written to XML and stored along with its current state.

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module ports

- Modules are connected using ports.
- A port is an address identified by either
  1. a symbolic name (‘freq’)
  2. a numeric identifier
- Each module instance may be identified by either
  1. a symbolic name (‘sinus1’)
  2. a globally unique numeric id
- A port may be addressed by
  1. its symbolic name:

     ```
     /sinus1/freq
     ```

  2. or by its UID/Port ID combination:

     ```
     123 1
     ```

Integra

Module connection commands

Command	Purpose
/load <module-name>	Instantiate a module in a given target
/remove <module id>	Remove a module instance
/connect <module id> <port number> <module id> <port number>	Connect two ports
/disconnect <module id> <port number> <module id> <port number>	Disconnect two ports
/send <module id> <port number> <value>	Send a value to a port

Integra

Module connection commands

Command	Purpose
/load <module-name>	Instantiate a module in a given target
/remove <module id>	Remove a module instance
/connect <module id> <port number> <module id> <port number>	Connect two ports
/disconnect <module id> <port number> <module id> <port number>	Disconnect two ports
/send <module id> <port number> <value>	Send a value to a port

Integra

Module connection commands

Command	Purpose
/load <module-name>	Instantiate a module in a given target
/remove <module id>	Remove a module instance
/connect <module id> <port number> <module id> <port number>	Connect two ports
/disconnect <module id> <port number> <module id> <port number>	Disconnect two ports
/send <module id> <port number> <value>	Send a value to a port

Integra

Module connection commands

Command	Purpose
/load <module-name>	Instantiate a module in a given target
/remove <module id>	Remove a module instance
/connect <module id> <port number> <module id> <port number>	Connect two ports
/disconnect <module id> <port number> <module id> <port number>	Disconnect two ports
/send <module id> <port number> <value>	Send a value to a port

Integra

Module connection commands

Command	Purpose
/load <module-name>	Instantiate a module in a given target
/remove <module id>	Remove a module instance
/connect <module id> <port number> <module id> <port number>	Connect two ports
/disconnect <module id> <port number> <module id> <port number>	Disconnect two ports
/send <module id> <port number> <value>	Send a value to a port

Integra

Properties of Integra ports

- No distinction between audio and control rate ports—the rate is defined for the connection
- No distinction between input and output ports
- Automatic data mapping between connected ports:
  - Unit conversion: midi note nr -> frequency
  - Type conversion
- A port is an address

Integra

Properties of Integra ports

- No distinction between audio and control rate ports—the rate is defined for the connection
- No distinction between input and output ports
- Automatic data mapping between connected ports:
  - Unit conversion: midi note nr -> frequency
  - Type conversion
- A port is an address

Integra

Properties of Integra ports

- No distinction between audio and control rate ports—the rate is defined for the connection
- No distinction between input and output ports
- Automatic data mapping between connected ports:
  - Unit conversion: midi note nr -> frequency
  - Type conversion
- A port is an address

Integra

Properties of Integra ports

- No distinction between audio and control rate ports—the rate is defined for the connection
- No distinction between input and output ports
- Automatic data mapping between connected ports:
  - Unit conversion: midi note nr -> frequency
  - Type conversion
- A port is an address

Integra

Properties of Integra ports

- No distinction between audio and control rate ports—the rate is defined for the connection
- No distinction between input and output ports
- Automatic data mapping between connected ports:
  - Unit conversion: midi note nr -> frequency
  - Type conversion
- A port is an address

Integra

Properties of Integra ports

- No distinction between audio and control rate ports—the rate is defined for the connection
- No distinction between input and output ports
- Automatic data mapping between connected ports:
  - Unit conversion: midi note nr -> frequency
  - Type conversion
- A port is an address

Integra

Interfacing Integra

A cross platform shared library

libIntegra provides application developers with the functionality to read, write and validate Integra compliant XML.
Provides bindings to a variety of languages e.g. Python.

Integra database

Provides persistent storage
Postgresql ORDMS retaining module inheritance.

Database UI

Provides online access to the DB.
Users may browse, add, edit, and download modules through the DBUI.

Integra

Interfacing Integra

A cross platform shared library

libIntegra provides application developers with the functionality to read, write and validate Integra compliant XML.
Provides bindings to a variety of languages e.g. Python.

Integra database

Provides persistent storage
Postgresql ORDMS retaining module inheritance.

Database UI

Provides online access to the DB.
Users may browse, add, edit, and download modules through the DBUI.

Integra

Interfacing Integra

A cross platform shared library

libIntegra provides application developers with the functionality to read, write and validate Integra compliant XML.
Provides bindings to a variety of languages e.g. Python.

Integra database

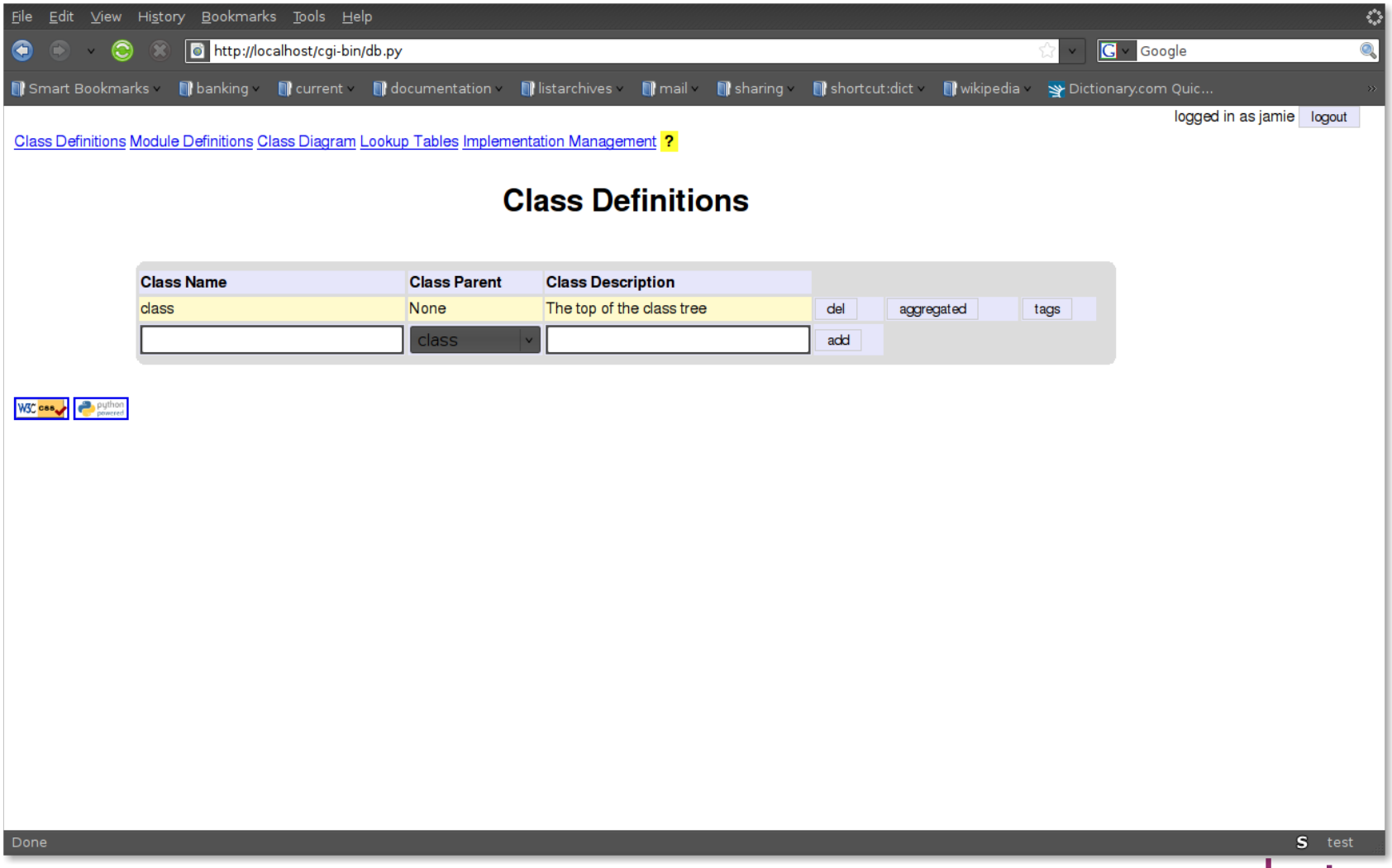
Provides persistent storage
Postgresql ORDMS retaining module inheritance.

Database UI

Provides online access to the DB.
Users may browse, add, edit, and download modules through the DBUI.

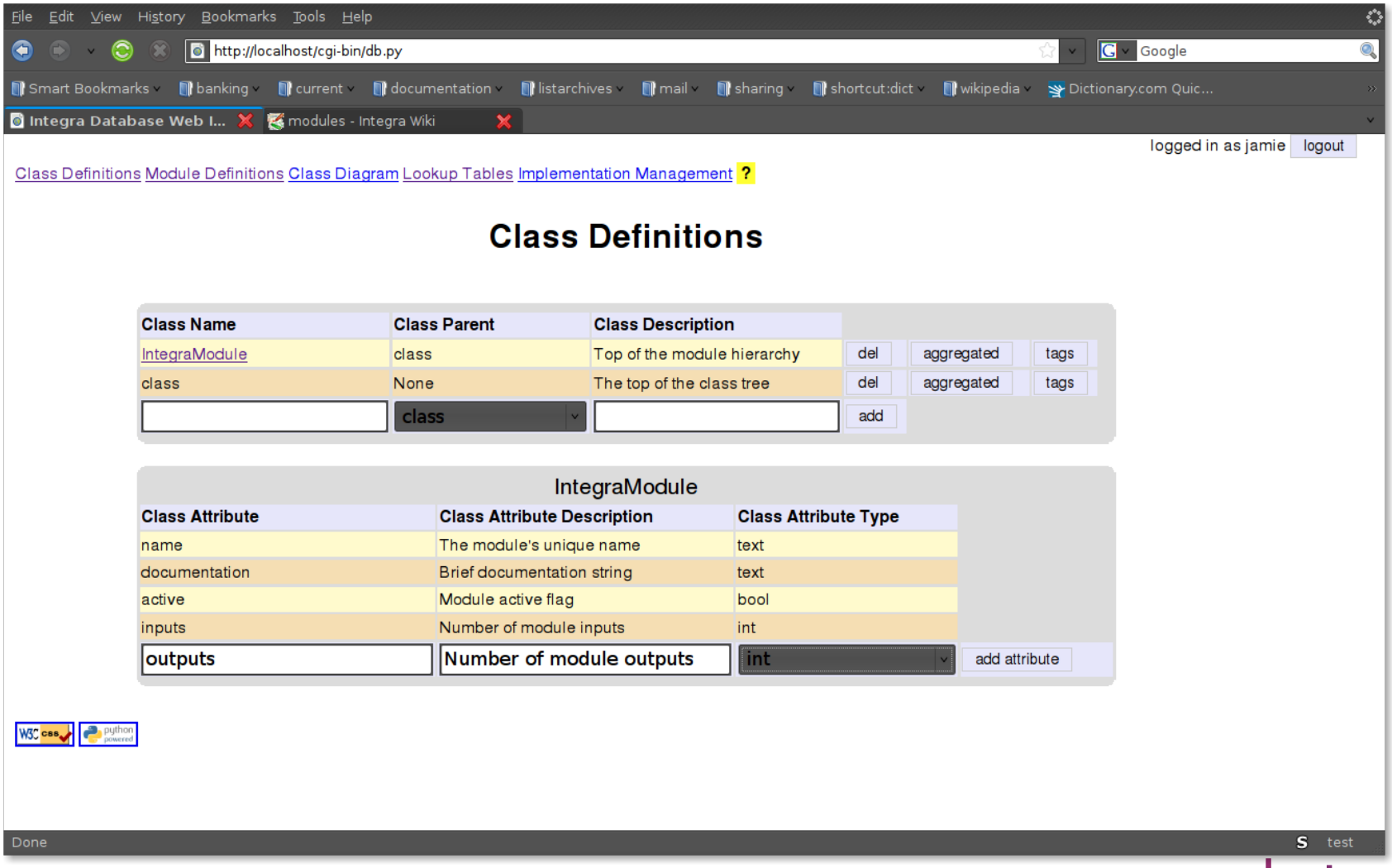
Integra

Database UI

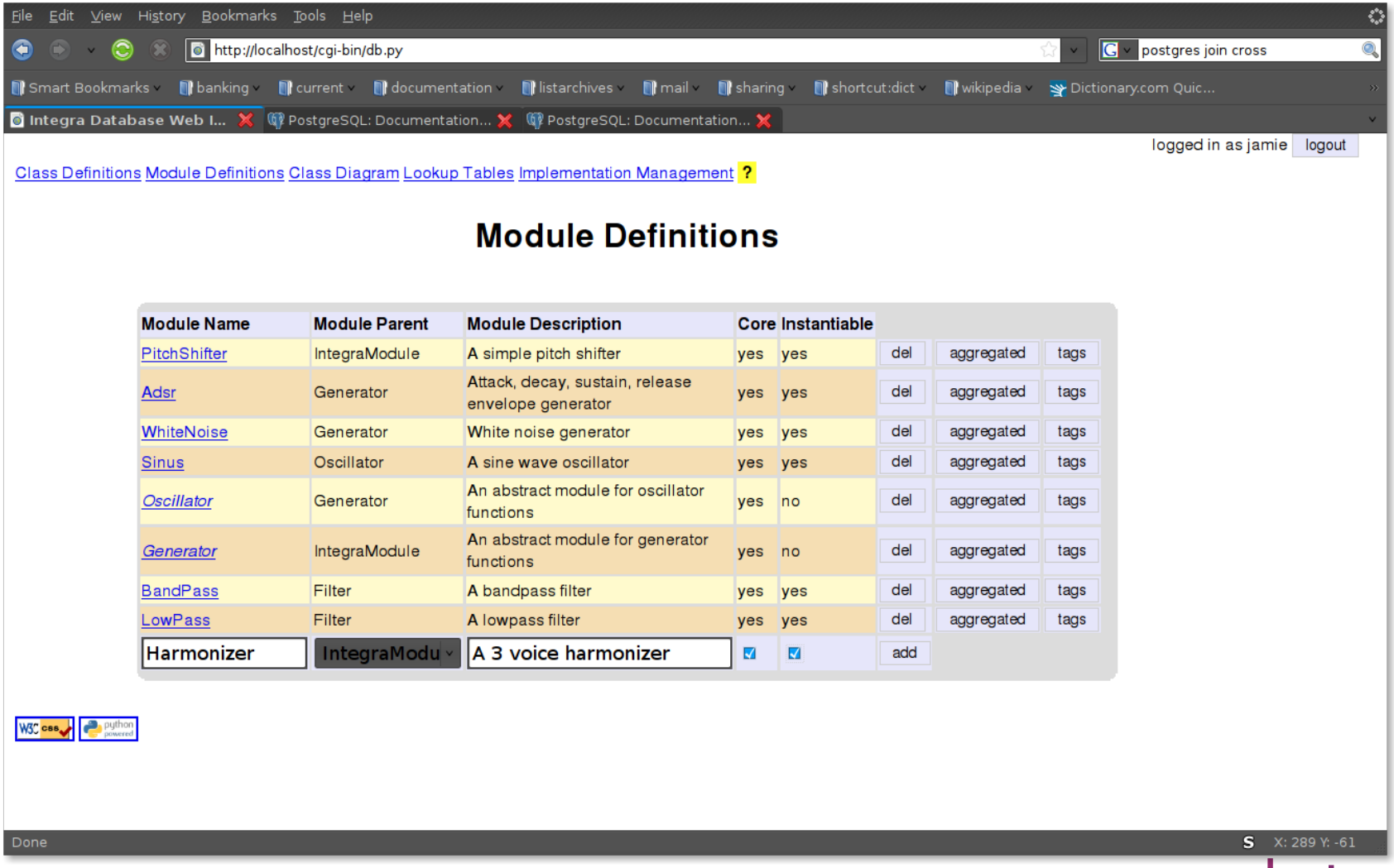


Integra

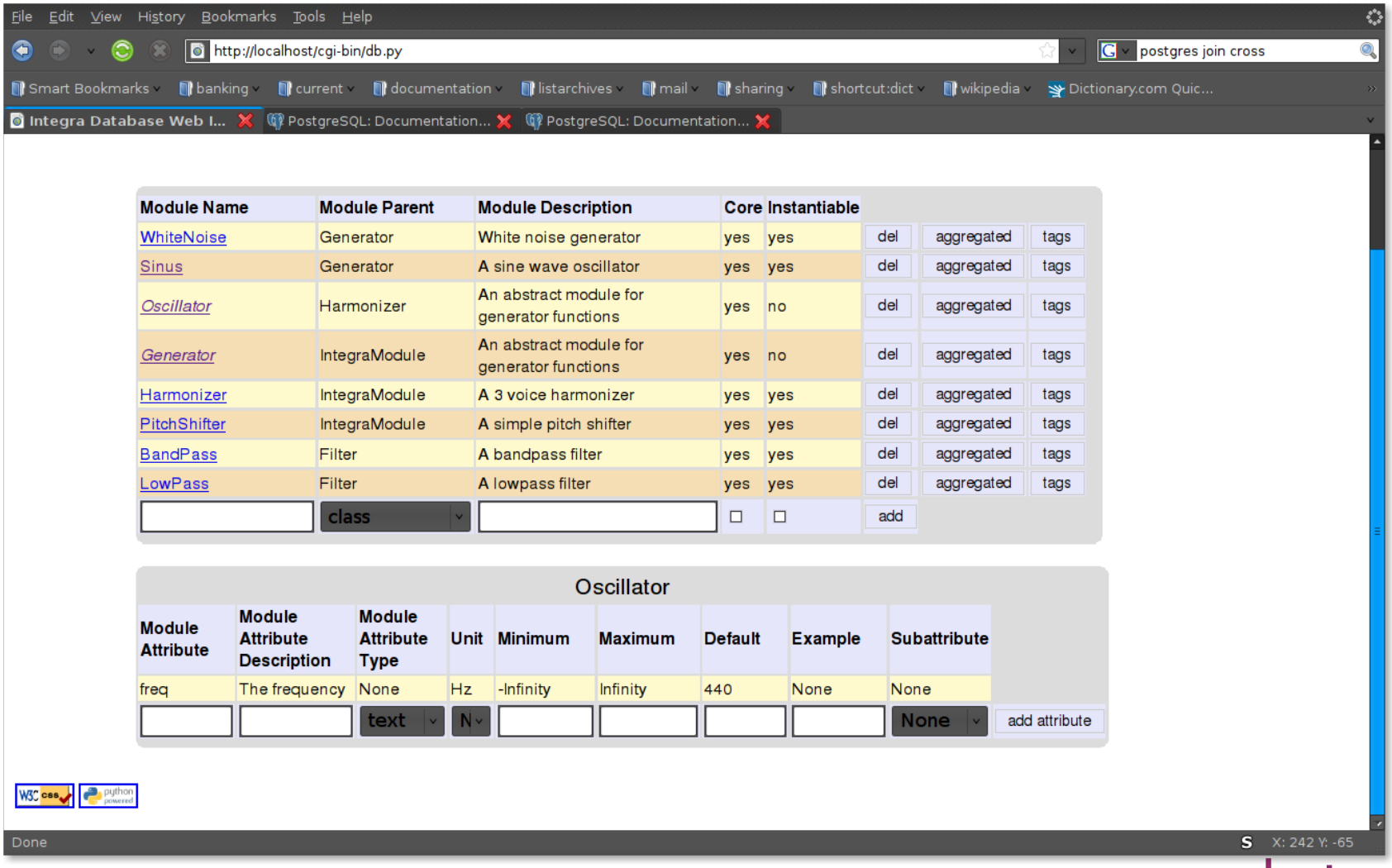
Database UI


File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/db.py
Google
Smart Bookmarks banking current documentation listarchives mail sharing shortcut:dict wikipedia Dictionary.com Quic...
Integra Database Web I...
modules - Integra Wiki
logged in as jamie logout
Class Definitions Module Definitions Class Diagram Lookup Tables Implementation Management ?
Class Definitions
Class Name
Class Parent
Class Description
IntegraModule
class
Top of the module hierarchy
del
aggregated
tags
class
None
The top of the class tree
del
aggregated
tags
class
add
IntegraModule
Class Attribute
Class Attribute Description
Class Attribute Type
name
The module's unique name
text
documentation
Brief documentation string
text
active
Module active flag
bool
inputs
Number of module inputs
int
outputs
Number of module outputs
int
add attribute
Done
test

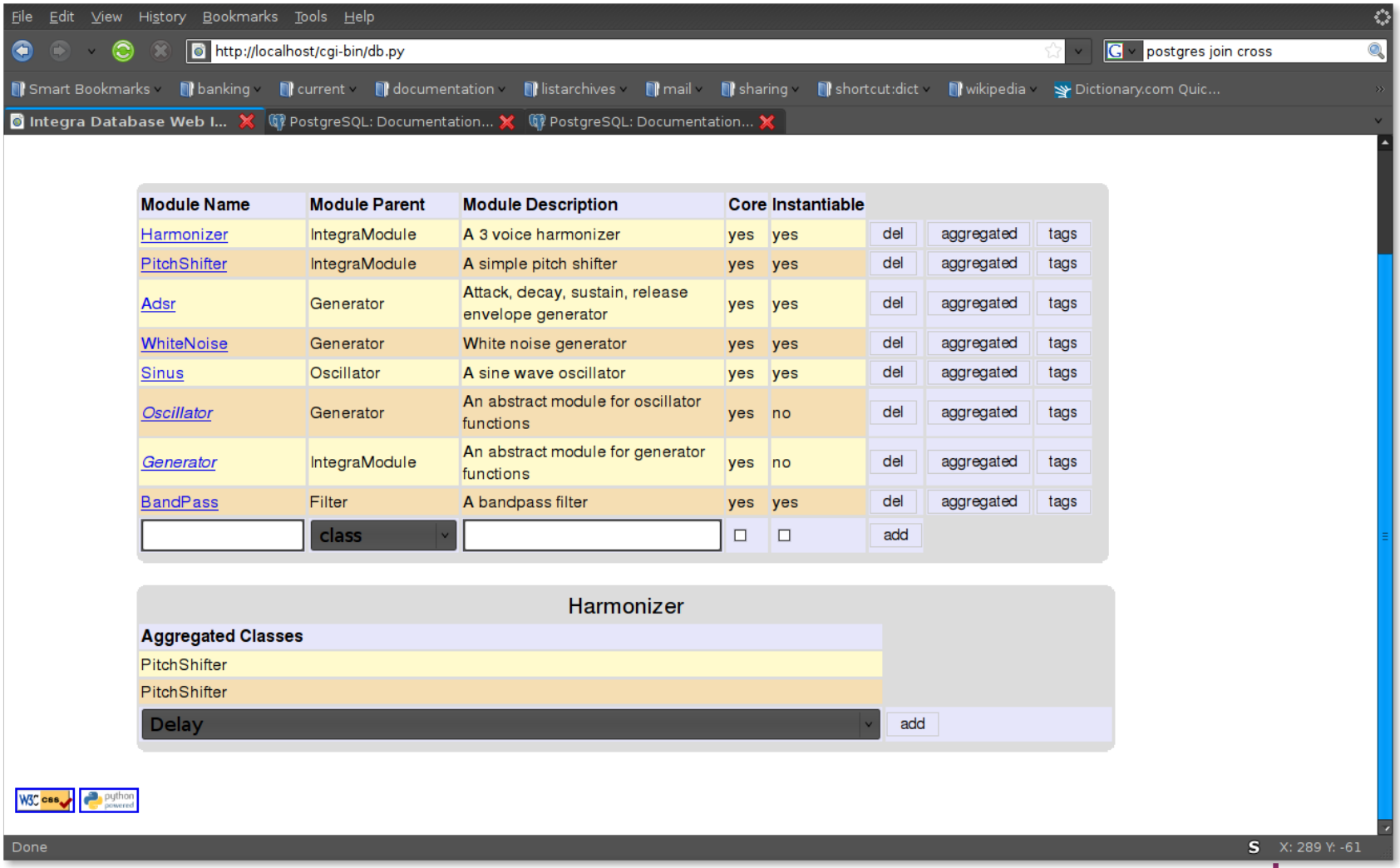
Database UI


File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/db.py
postgres join cross
Smart Bookmarks banking current documentation listarchives mail sharing shortcut:dict wikipedia Dictionary.com Quic...
Integra Database Web I...
PostgreSQL: Documentation...
PostgreSQL: Documentation...
logged in as jamie logout
Class Definitions Module Definitions Class Diagram Lookup Tables Implementation Management ?
Module Definitions
Module Name | Module Parent | Module Description | Core | Instantiable
PitchShifter | IntegraModule | A simple pitch shifter | yes | yes | del | aggregated | tags
Adsr | Generator | Attack, decay, sustain, release envelope generator | yes | yes | del | aggregated | tags
WhiteNoise | Generator | White noise generator | yes | yes | del | aggregated | tags
Sinus | Oscillator | A sine wave oscillator | yes | yes | del | aggregated | tags
Oscillator | Generator | An abstract module for oscillator functions | yes | no | del | aggregated | tags
Generator | IntegraModule | An abstract module for generator functions | yes | no | del | aggregated | tags
BandPass | Filter | A bandpass filter | yes | yes | del | aggregated | tags
LowPass | Filter | A lowpass filter | yes | yes | del | aggregated | tags
Harmonizer | IntegraModu | A 3 voice harmonizer | add
Done
X: 289 Y: -61

Database UI

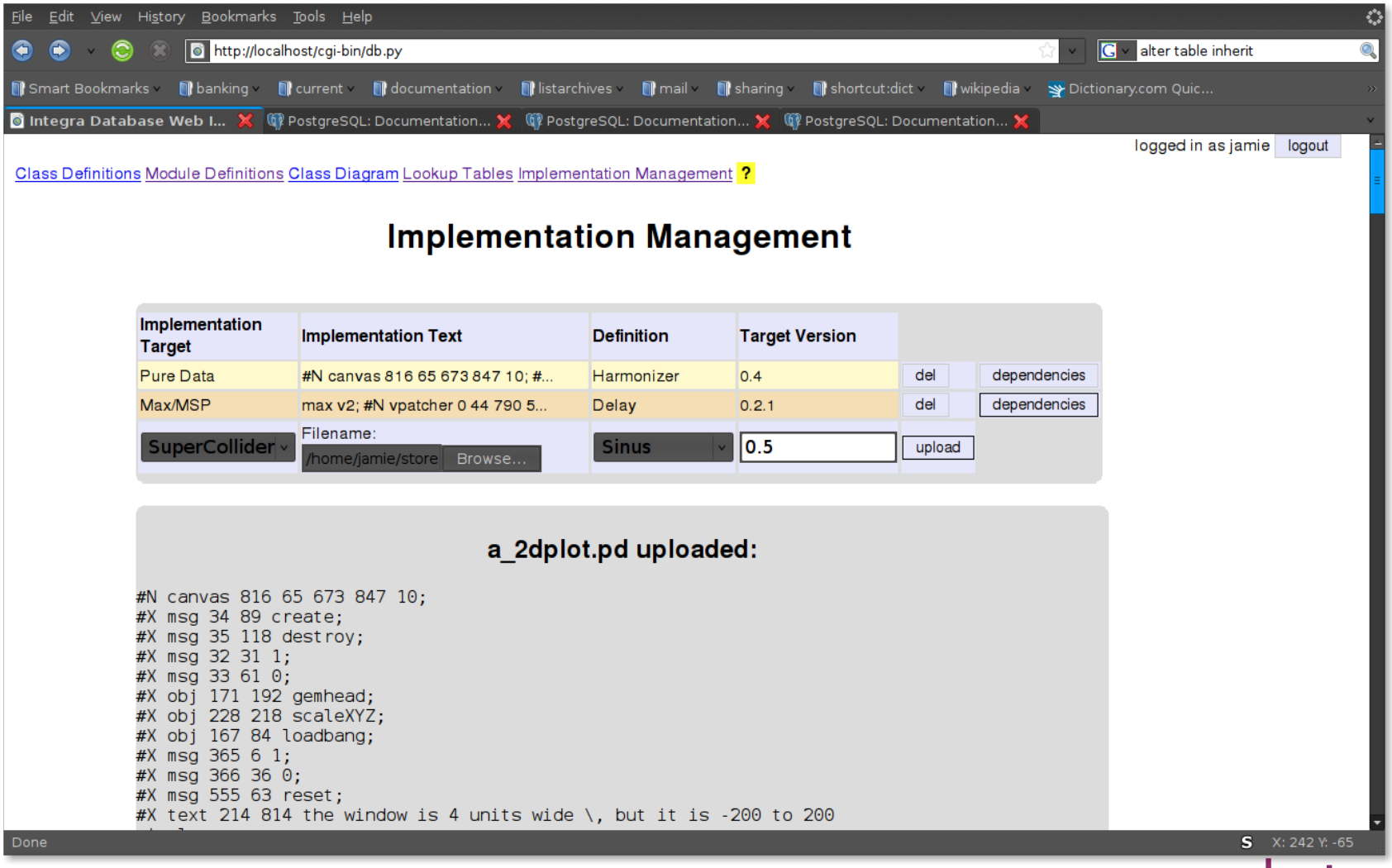

File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/db.py
postgres join cross
Smart Bookmarks banking current documentation listarchives mail sharing shortcut:dict wikipedia Dictionary.com Quic...
Integra Database Web I...
PostgreSQL: Documentation...
PostgreSQL: Documentation...
Module Name | Module Parent | Module Description | Core | Instantiable
WhiteNoise | Generator | White noise generator | yes | yes | del | aggregated | tags
Sinus | Generator | A sine wave oscillator | yes | yes | del | aggregated | tags
Oscillator | Harmonizer | An abstract module for generator functions | yes | no | del | aggregated | tags
Generator | IntegraModule | An abstract module for generator functions | yes | no | del | aggregated | tags
Harmonizer | IntegraModule | A 3 voice harmonizer | yes | yes | del | aggregated | tags
PitchShifter | IntegraModule | A simple pitch shifter | yes | yes | del | aggregated | tags
BandPass | Filter | A bandpass filter | yes | yes | del | aggregated | tags
LowPass | Filter | A lowpass filter | yes | yes | del | aggregated | tags
class
add
Oscillator
Module Attribute | Module Attribute Description | Module Attribute Type | Unit | Minimum | Maximum | Default | Example | Subattribute
freq | The frequency | None | Hz | -Infinity | Infinity | 440 | None | None
text
None
add attribute
Done
X: 242 Y: -65

Database UI



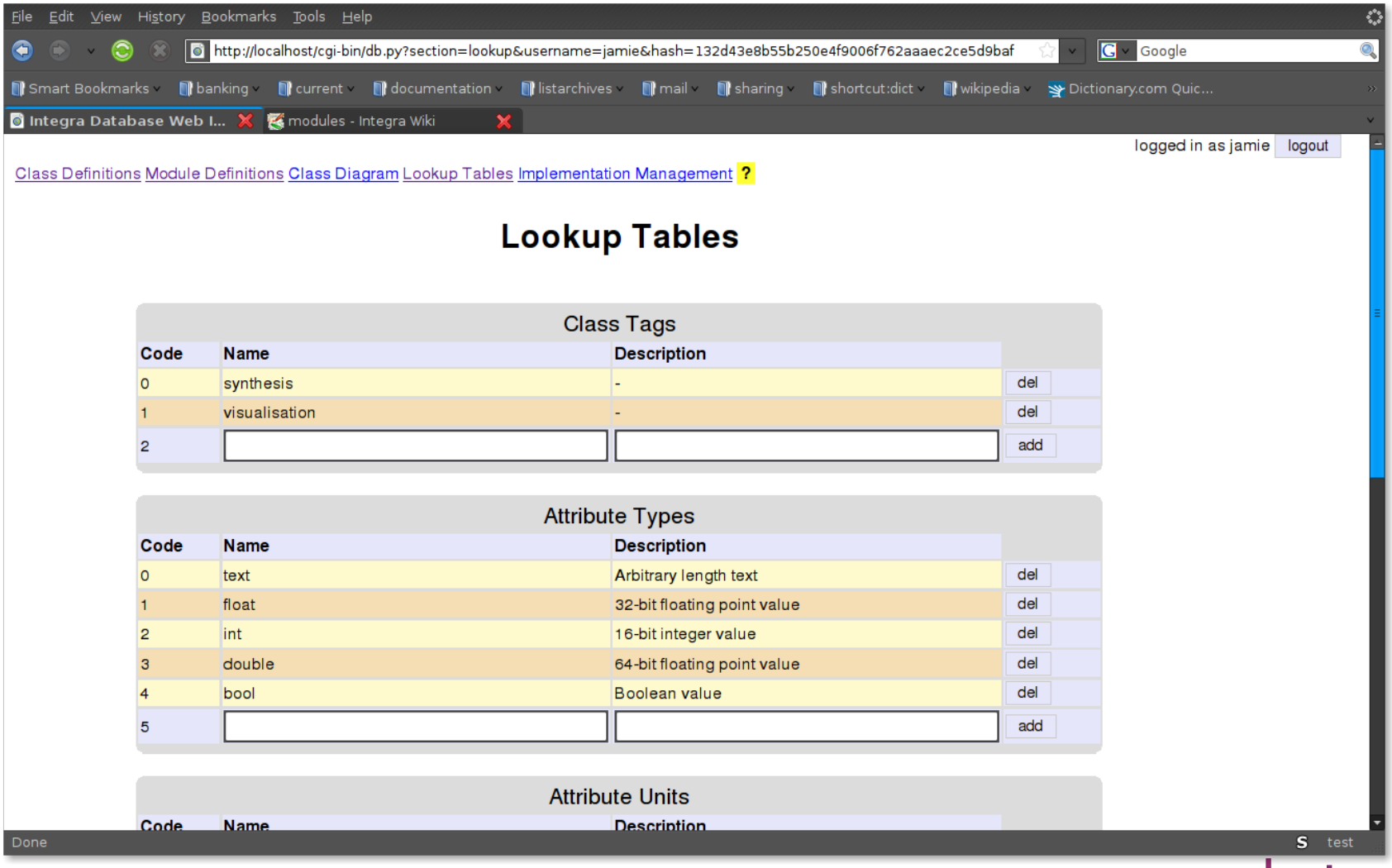
Integra

Database UI

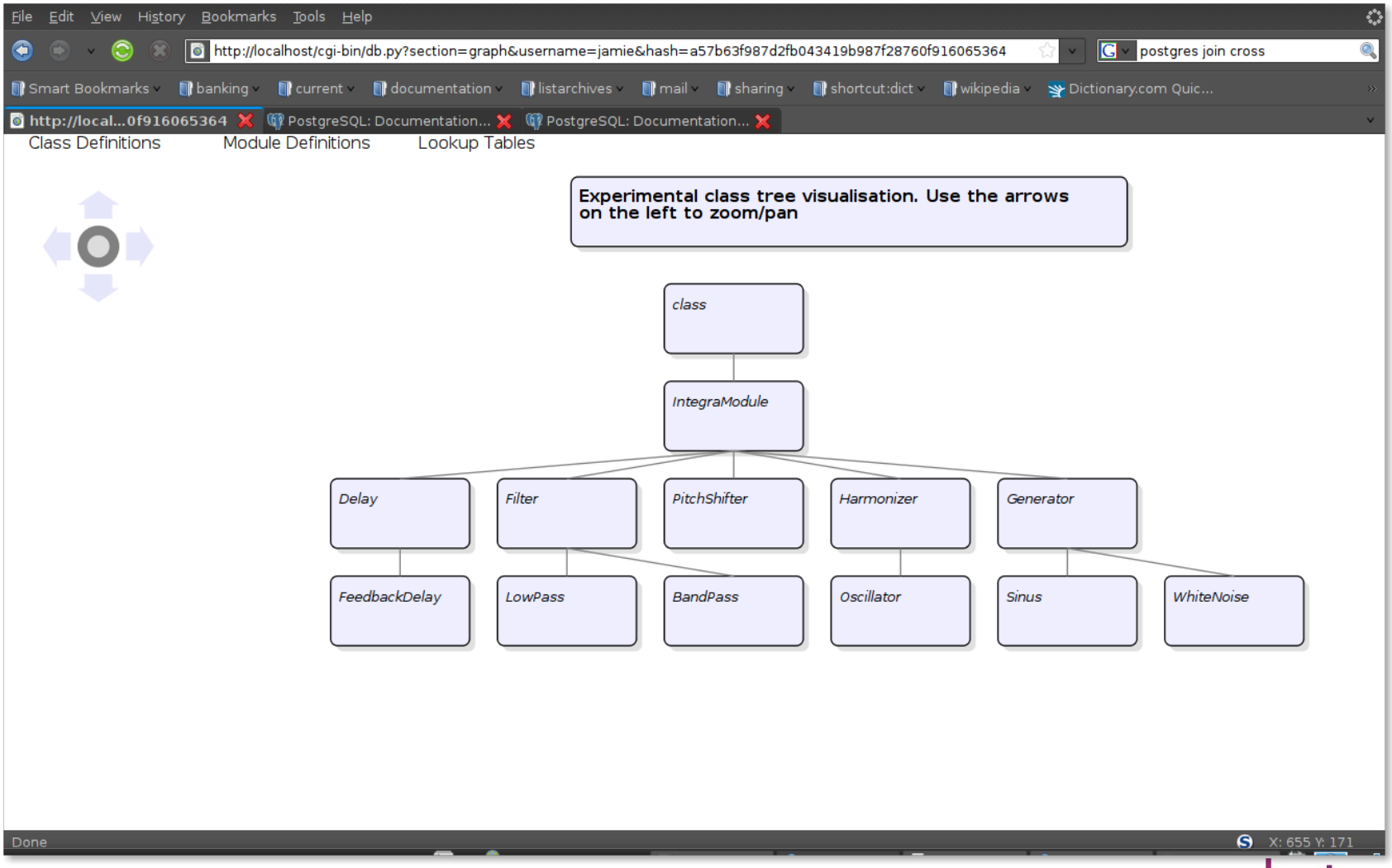


Integra

Database UI


File Edit View History Bookmarks Tools Help
http://localhost/cgi-bin/db.py?section=lookup&username=jamie&hash=132d43e8b55b250e4f9006f762aaaec2ce5d9baf
Google
Smart Bookmarks banking current documentation listarchives mail sharing shortcut:dict wikipedia Dictionary.com Quic...
Integra Database Web I...
modules - Integra Wiki
logged in as jamie
logout
Class Definitions Module Definitions Class Diagram Lookup Tables Implementation Management ?
Lookup Tables
Class Tags
Code Name Description
0 synthesis - del
1 visualisation - del
2 add
Attribute Types
Code Name Description
0 text Arbitrary length text del
1 float 32-bit floating point value del
2 int 16-bit integer value del
3 double 64-bit floating point value del
4 bool Boolean value del
5 add
Attribute Units
Code Name Description
Done
test

Database UI



Integra

Use case examples (1:2)

Integra projects

Migrated works

Works migrated within the Integra project serve as support for continuing development of the framework.

Madonna of Winter and Spring

by Jonathan Harvey, ported to Integra.

- Development of Pd based DSSI plugin host
- Hexter: A DX7 emulation plugin.
- Allows emulation of the TX816, DX1 and DX7 and makes these addressable via the Integra framework.

The Integra GUI

Provide a powerful but simple interface giving non-computer savvy musicians and composers access to live electronics.
Interfaces the libIntegra and the database.

Use case examples (1:2)

Integra projects

Migrated works

Works migrated within the Integra project serve as support for continuing development of the framework.

Madonna of Winter and Spring

by Jonathan Harvey, ported to Integra.

- Development of Pd based DSSI plugin host
- Hexter: A DX7 emulation plugin.
- Allows emulation of the TX816, DX1 and DX7 and makes these addressable via the Integra framework.

The Integra GUI

Provide a powerful but simple interface giving non-computer savvy musicians and composers access to live electronics.
Interfaces the libIntegra and the database.

Use case examples (1:2)

Integra projects

Migrated works

Works migrated within the Integra project serve as support for continuing development of the framework.

Madonna of Winter and Spring

by Jonathan Harvey, ported to Integra.

- Development of Pd based DSSI plugin host
- Hexter: A DX7 emulation plugin.
- Allows emulation of the TX816, DX1 and DX7 and makes these addressable via the Integra framework.

The Integra GUI

Provide a powerful but simple interface giving non-computer savvy musicians and composers access to live electronics.
Interfaces the libIntegra and the database.

Use case examples (1:2)

Integra projects

Migrated works

Works migrated within the Integra project serve as support for continuing development of the framework.

Madonna of Winter and Spring

by Jonathan Harvey, ported to Integra.

- Development of Pd based DSSI plugin host
- Hexter: A DX7 emulation plugin.
- Allows emulation of the TX816, DX1 and DX7 and makes these addressable via the Integra framework.

The Integra GUI

Provide a powerful but simple interface giving non-computer savvy musicians and composers access to live electronics.
Interfaces the libIntegra and the database.

Use case examples (1:2)

Integra projects

Migrated works

Works migrated within the Integra project serve as support for continuing development of the framework.

Madonna of Winter and Spring

by Jonathan Harvey, ported to Integra.

- Development of Pd based DSSI plugin host
- Hexter: A DX7 emulation plugin.
- Allows emulation of the TX816, DX1 and DX7 and makes these addressable via the Integra framework.

The Integra GUI

Provide a powerful but simple interface giving non-computer savvy musicians and composers access to live electronics.
Interfaces the libIntegra and the database.

Use case examples (1:2)

Integra projects

Migrated works

Works migrated within the Integra project serve as support for continuing development of the framework.

Madonna of Winter and Spring

by Jonathan Harvey, ported to Integra.

- Development of Pd based DSSI plugin host
- Hexter: A DX7 emulation plugin.
- Allows emulation of the TX816, DX1 and DX7 and makes these addressable via the Integra framework.

The Integra GUI

Provide a powerful but simple interface giving non-computer savvy musicians and composers access to live electronics. Interfaces the libIntegra and the database.

Use case examples (2:2)

Auxiliary projects

Sonar 2D

libIntegra forms the back end for the Sonar 2D application by Bullock.

The annotated score

libIntegra, the Integra XML file format and a web app under development forms the basis for the *annotated score*: A user adaptable, versioned, “open-sourced” musical score documenting all versions of a musical work, including meta-information.

Integra

Use case examples (2:2)

Auxiliary projects

Sonar 2D

libIntegra forms the back end for the Sonar 2D application by Bullock.

The annotated score

libIntegra, the Integra XML file format and a web app under development forms the basis for the *annotated score*: A user adaptable, versioned, “open-sourced” musical score documenting all versions of a musical work, including meta-information.

Integra

Project status

- Hosted on Sourceforge: `http://sourceforge.net`
- Small but active developer community
- libIntegra is in pre-alpha development

Future work

- Populate the database
- Explore ways of storing implementation specific data in a software neutral manner.
  - By creating a set of implementation primitives
  - Create a scripting language, perhaps along the ATOM

Project status

- Hosted on Sourceforge: `http://sourceforge.net`
- Small but active developer community
- libIntegra is in pre-alpha development

Future work

- Populate the database
- Explore ways of storing implementation specific data in a software neutral manner.
  - By creating a set of implementation primitives
  - Create a scripting language, perhaps atop the XML

Project status

- Hosted on Sourceforge: `http://sourceforge.net`
- Small but active developer community
- libIntegra is in pre-alpha development

Future work

- Populate the database
- Explore ways of storing implementation specific data in a software neutral manner.
  - By creating a set of implementation primitives
  - Create a scripting language, perhaps atop the XML

Project status

- Hosted on Sourceforge: `http://sourceforge.net`
- Small but active developer community
- libIntegra is in pre-alpha development

Future work

- Populate the database
- Explore ways of storing implementation specific data in a software neutral manner.
  - By creating a set of implementation primitives.
  - Create a scripting language, perhaps alongside FAUST.

Integra

Project status

- Hosted on Sourceforge: `http://sourceforge.net`
- Small but active developer community
- libIntegra is in pre-alpha development

Future work

- Populate the database
- Explore ways of storing implementation specific data in a software neutral manner.
  - By creating a set of implementation primitives.
  - Create a scripting language, perhaps alongside FAUST.

Integra

Project status

- Hosted on Sourceforge: `http://sourceforge.net`
- Small but active developer community
- libIntegra is in pre-alpha development

Future work

- Populate the database
- Explore ways of storing implementation specific data in a software neutral manner.
  - By creating a set of implementation primitives.
  - Create a scripting language, perhaps alongside FAUST.

Integra

Project status

- Hosted on Sourceforge: `http://sourceforge.net`
- Small but active developer community
- libIntegra is in pre-alpha development

Future work

- Populate the database
- Explore ways of storing implementation specific data in a software neutral manner.
  - By creating a set of implementation primitives.
  - Create a scripting language, perhaps alongside FAUST.

Integra

Project status

- Hosted on Sourceforge: `http://sourceforge.net`
- Small but active developer community
- libIntegra is in pre-alpha development

Future work

- Populate the database
- Explore ways of storing implementation specific data in a software neutral manner.
  - By creating a set of implementation primitives.
  - Create a scripting language, perhaps alongside FAUST.

Integra

Conclusion

XML format: IXD

A new XML-based format for persistently storing data relating to musical works, in particular works including live electronics.

libIntegra

A library allowing users to access data in the XML.

Database

An object-relational database used for storing and versioning of IXD data.

Usability

The aim of the Integra project is to improve the usability, and to provide a robust mechanism for the sustainability of musical works. libIntegra and its file formats should ultimately provide a foundation for this.

Integra

Conclusion

XML format: IXD

A new XML-based format for persistently storing data relating to musical works, in particular works including live electronics.

libIntegra

A library allowing users to access data in the XML.

Database

An object-relational database used for storing and versioning of IXD data.

Usability

The aim of the Integra project is to improve the usability, and to provide a robust mechanism for the sustainability of musical works. libIntegra and its file formats should ultimately provide a foundation for this.

Integra

Conclusion

XML format: IXD

A new XML-based format for persistently storing data relating to musical works, in particular works including live electronics.

libIntegra

A library allowing users to access data in the XML.

Database

An object-relational database used for storing and versioning of IXD data.

Usability

The aim of the Integra project is to improve the usability, and to provide a robust mechanism for the sustainability of musical works. libIntegra and its file formats should ultimately provide a foundation for this.

Integra

Conclusion

XML format: IXD

A new XML-based format for persistently storing data relating to musical works, in particular works including live electronics.

libIntegra

A library allowing users to access data in the XML.

Database

An object-relational database used for storing and versioning of IXD data.

Usability

The aim of the Integra project is to improve the usability, and to provide a robust mechanism for the sustainability of musical works. libIntegra and its file formats should ultimately provide a foundation for this.

Integra

Thank you!

We wish to thank the organizers of the Melecon 2008 Conference for giving us this opportunity to present part of the Integra project.

Funding

The Integra project is funded by the EC Culture 2000 and is a collaboration between Universities, research centers and New Music Ensembles in Europe.

Integra