Stencila Documentation

Release 0.1.0

Nokome Bentley

Sep 07, 2017

Contents

Contents:

CHAPTER 1

PythonContext

class stencila.**PythonContext**(*dir=None*)

runCode(*code*)
Run Python code

callCode(*code*, *inputs={}*)
Call Python code

spec = {'aliases': ['py', 'python'], 'base': 'Context', 'name': 'PythonContext'}

CHAPTER 2

SqliteContext

class stencila.**SqliteContext**(*dir=None*)

runCode(*code*)

callCode(*code*, *inputs={}*)

spec = {'aliases': ['sql', 'sqlite'], 'base': 'Context', 'name': 'SqliteContext'}

CHAPTER 3

Host

class stencila.**Host**

A *Host* allows you to create, get, run methods of, and delete instances of various types. The types can be thought of a "services" provided by the host e.g. *NoteContext*, *FilesystemStorer*

The API of a host is similar to that of a HTTP server. It's methods names (e.g. *post*, *get*) are similar to HTTP methods (e.g. *POST*, *GET*) but the sematics sometimes differ (e.g. a host's *put()* method is used to call an instance method)

A *Host* is not limited to beng served by HTTP and it's methods are exposed by both *HostHttpServer* and *HostWebsocketServer*. Those other classes are responsible for tasks associated with their communication protocol (e.g. serialising and deserialising objects).

This is a singleton class. There should only ever be one *Host* in memory in each process (although, for purposes of testing, this is not enforced)

id

user_dir()

Get the current user's Stencila data directory.

This is the directory that Stencila configuration settings, such as the installed Stencila hosts, and document buffers get stored.

temp_dir()

Get the current Stencila temporary directory

environ()

Get the environment of this host including the version of Node.js and versions of installed packages (local and globals)

Returns The environment as a dictionary of dictionaries

manifest()

Get a manifest for this host

The manifest describes the host and it's capabilities. It is used by peer hosts to determine which "types" this host provides and which "instances" have already been instantiated.

Returns A manifest object

install()
Installation of a host involves creating a file *py.json* inside of the user's Stencila data (see *user_dir()*) directory which describes the capabilities of this host.

post(*type*, *name=None*, *options={}*)
Create a new instance of a type

Parameters

- **type** – Type of instance
- **name** – Name of new instance
- **options** – Additional arguments to pass to constructor

Returns Address of newly created instance

get(*id*)
Get an instance

Parameters **id** – ID of instance

Returns The instance

put(*id*, *method*, *kwargs={}*)
Call a method of an instance

Parameters

- **id** – ID of instance
- **method** – Name of instance method
- **kwargs** – A dictionary of method arguments

Returns Result of method call

delete(*id*)
Delete an instance

Parameters **id** – ID of instance

start(*address='127.0.0.1'*, *port=2000*, *authorization=True*, *quiet=False*)
Start serving this host

Currently, HTTP is the only server available for hosts. We plan to implement a *HostWebsocketServer* soon.

Returns self

stop(*quiet=False*)
Stop serving this host

Returns self

run(*address='127.0.0.1'*, *port=2000*, *authorization=True*, *quiet=False*, *echo=False*)
Start serving this host and wait for connections indefinitely

servers

urls

view()
View this host in the browser

Opens the default browser at the URL of this host

CHAPTER 4

HostHttpServer

class stencila.**HostHttpServer**(*host*, *address='127.0.0.1'*, *port=2000*, *authorization=True*)

url
Get the URL for this server

start(*real=True*)
Start the server

stop(*real=True*)
Stop the server

handle(*request*)
Handle a HTTP request

route(*verb*, *path*)
Route a HTTP request

options(*request*)

home(*request*)

static(*request*, *path*)

post(*request*, *type*)

get(*request*, *id*)

put(*request*, *id*, *method*)

delete(*request*, *id*)

ticket_create()
Create a ticket (a single-use access token)

ticket_check(*ticket*)
Check that a ticket is valid. * If it is, then it is removed from the list of tickets and *true* is returned. Otherwise, returns *false*

ticketed_url()
Create a URL with a ticket query parameter so users can connect to this server

token_create()
Create a token (a multiple-use access token)

token_check(*token*)
Check that a token is valid.

CHAPTER 5

Value

A module for packing and unpacking values so that they can be transferred between languages and hosts.

stencila.value.**type**(*value*)
Get the type code for a value

Parameters **value** – A Python value

Returns Type code for value

stencila.value.**pack**(*value*)
Pack an object into a value package

Parameters **value** – A Python value

Returns A value package

stencila.value.**unpack**(*pkg*)
Unpack a value package into a Python value

Parameters **pkg** – The value package

Returns A Python value

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index