ITK Design

model

Author: Jesper Kristensen

Date: 2014-08-13

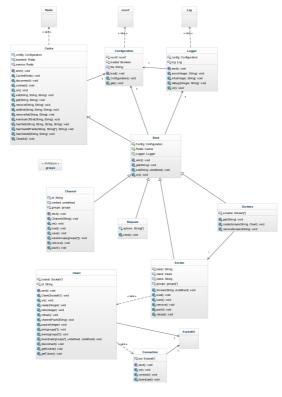
Table of Contents

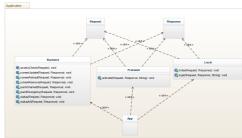
Ove	erview	_		 -	 -	 -	 -			 	 _
	Model description										
	Diagrams										
Classifiers		_									
	Base										
	Configuration										
	Cache										
	nconf										
	Logger										
	Log										
	Redis										
	Channel										
	Client										
	Connection										
	Request										
	Screen										
	Screens										
	SocketIO										
Pac	ckages	_								 	
	Application										
	Арр										
	Backend										
	Frontend										
	Local										
	Request										
	Responce										

1. Overview

1.1 Model Description

1.2 Diagrams





2 Classifiers

2.1 Class Base

Abstract class to hold basic functions used in all inherited classes. Basic function to set/get properties and configuration.

2.1.1 Attributes

- Config : Configuration [1]

- Redis: Cache [1]

- Logger : Logger [1]

2.1.2 Generalizations

None

2.1.3 Operations

- Void emit ()

- Void get (property : String)

null

- Void on ()

2.1.4 Associations

- configuration : Configuration [1]

- logger : Logger [1]

- cache : Cache [1]

2.2 Class Configuration

Wrapper class to get configuration form the config.json file.

This is a singleton class to prevent loading the configuration more than once.

2.2.1 Attributes

- **nconf** : nconf [1]

Description: NConf module requirement.

- loaded : Boolean [1]

Description: True if the configuration file is loaded else false.

- file : String [1]

Description: Name of the configuration file.

2.2.2 Generalizations

None

2.2.3 Operations

- Void load ()

Description: Loads configuration from the filesystem.

- Void Configuration ()

Description: Default constructor that loads configuration file(s) on initialization.

- Void get ()

Description: Get configuration option as Object or String, if it's a single value.

2.3 Class Cache

Cache is used to speed up the communication with the frontend. It stores information from the backend and stores information from the frontend.

It's a singleton class to ensure that only on connection is created to the backend cache store.

2.3.1 Attributes

- config : Configuration [1]

Description: Loaded configuration object.

- backend : Redis [1]

Description: The backend end module or cache storage used.

- service : Redis [1]

Description: Connection to the backend storage used to store data that should be

cached.

2.3.2 Generalizations

None

2.3.3 Operations

- Void **emit** ()

Description: Emit an event.

- Void Cache (backend : Redis)

Description: Default constructor for the class.

- Void disconnect ()

Description: Disconnect from the Redis server thereby freeing the connection.

- Void connect ()

Description: Create a connection to the Redis server.

- Void **on** ()

Description: Listen to event emitted from inside the object.

- Void **set** (key : String, value : String, callback : String)

Description: Set a simple string into the cache stored on the key given.

- Void **get** (key : String, callback : String)

Description: Get simple string from the cache indexed by the key given. When the string is retrieved the callback will be called.

- Void **remove** (key : String, callback : String)

Description: Remove simple string from the cache indexed by the key given. When the string is removed the callback will be called.

- Void **addSet** (key: String, value: String, callback: String)

Description: Add a string to a set indexed by key. If the key does not exist, a new set is created before adding the specified value.

When the string has be added to the set the callback is called.

- Void **removeSet** (key : String, value : String)

Description: Remove a string/member from a set indexed by key. When the string has be removed the callback is called.

- Void **membersOfSet** (key: String, callback: String)

Description: Returns all the strings of the set stored at the key.

- Void **hashSet** (key: String, hash: String, value: String, callback: String)

Description: Sets the value in a hash stored index by key to the value. If key does not exist, a new key holding a hash is created.

- Void **hashGetAllFields** (key : String, hashes : String, callback : String)

Description: Get all values/fields store in an hash based on "hashes" given in the parameter.

- Void hashGetAll (key: String, callback: String)

Description: Get all values in a hash.

- Void ClearAll ()

Description: Clears all cache entries.

2.3.4 Associations

- configuration : Configuration [1]

2.4 Class nconf

This is the NPM package that handles the config.js configuration file.

2.4.1 Attributes

No additional attributes

2.4.2 Generalizations

None

2.4.3 Operations

No additional operations

2.5 Class Logger

2.5.1 Attributes

- **config** : Configuration [1]

- log : Log [1]

2.5.2 Generalizations

None

2.5.3 Operations

- Void emit ()

- Void **error** (code : Integer, message : String)

- Void **info** (code : Integer, message : String)

- Void **debug** (code : Integer, message : String)

- Void **on** ()

2.5.4 Associations

- configuration : Configuration [1]

2.6 Class Log

This is the NPM package that handles logging to either the file system or other logging resources.

2.6.1 Attributes

No additional attributes

2.6.2 Generalizations

None

2.6.3 Operations

No additional operations

2.7 Class Redis

This is the NPM package that handles communication with the Redis value/key store server.

2.7.1 Attributes

No additional attributes

2.7.2 Generalizations

None

2.7.3 Operations

No additional operations

2.8 Class Channel

2.8.1 Attributes

- id : String [1]

null

- groups : groups [1]

2.8.2 Generalizations

- Base

2.8.3 Operations

- Void **emit** ()

- Void Channel (channelID : String)

- Void **on** ()

- Void load ()	
- Void save ()	
- Void saveGroups (groups : groups)	
- Void remove ()	
- Void push ()	
2.9 Class Client	
2.9.1 Attributes	
- socket : SocketIO [1]	
- id : String [1]	
2.9.2 Generalizations None	
2.9.3 Operations	
- Void emit ()	
- Void Client (SocketID : SocketIO)	
- Void on ()	

- Void ready (code : Integer)	
- Void kick (code : Integer)	
- Void reload ()	
- Void channelPush (content : String)	
- Void pause (code : Integer)	
- Void join (groups : groups)	
- Void leave (groups : groups)	
null	
- Void disconnect ()	
- Void getSocket ()	
- Void getToken ()	
2.9.4 Associations	
- socketIO : SocketIO [1]	

2.10 Class Connection

2.10.1 Attributes

- sio : SocketIO [1]

2.10.2 Generalizations

None

2.10.3 Operations

- Void emit ()
- Void **on** ()
- Void connect ()
- Void boardcast ()

2.10.4 Associations

- socketIO : SocketIO [1]

2.11 Class Request

2.11.1 Attributes

- **options** : String [*]

2.11.2 Generalizations

- Base

2.11.3 Operations

- Void send ()

- Void send ()

2.12 Class Screen

2.12.1 Attributes

- token : String [1]

- client : Client [1]

- name : String [1]

- **groups** : groups [*]

2.12.2 Generalizations

- Base

2.12.3 Operations

null

- Void **load** ()
- Void save ()
- Void remove ()
- Void **push** ()

- Void reload ()

2.13 Class Screens

2.13.1 Attributes

- screens : Screen [*]

2.13.2 Generalizations

- Base

2.13.3 Operations

- Void **get** (token : String)

- Void createScreen (token : String, client : Client)

- Void **removeScreen** (token : String)

2.13.4 Associations

- screen : Screen [*]

2.14 Class SocketIO

2.14.1 Attributes

No additional attributes

2.14.2 Generalizations

None

2.14.3 Operations

No additional operations

3 Packages

3.1 Package Application

3.1.1 Class App

3.1.1.1 Attributes

No additional attributes

3.1.1.2 Generalizations

None

3.1.1.3 Operations

No additional operations

3.1.2 Class Backend

3.1.2.1 Attributes

No additional attributes

3.1.2.2 Generalizations

None

3.1.2.3 Operations

- Void accessCheck (req : Request)
- Void **screenUpdate** (req : Request, res : Responce)
- Void **screenReload** (req : Request, res : Responce)
- Void **screenRemove** (req : Request, res : Responce)
- Void **pushChannel** (req : Request, res : Responce)
- Void **pushEmergency** (req : Request, res : Responce)

- Void **status** (req : Request, res : Responce)
- Void **statusAll** (req : Request, res : Responce)

3.1.3 Class Frontend

3.1.3.1 Attributes

No additional attributes

3.1.3.2 Generalizations

None

3.1.3.3 Operations

- Void **activate** (req : Request, res : Responce, jwt_secret : String)

3.1.4 Class Local

3.1.4.1 Attributes

No additional attributes

3.1.4.2 Generalizations

None

3.1.4.3 Operations

- Void **index** (req : Request, res : Responce)
- Void **login** (req : Request, res : Responce, jwt_secret : String)

3.1.5 Class Request

3.1.5.1 Attributes

3.1.5.2 Generalizations

None

3.1.5.3 Operations

No additional operations

3.1.6 Class Responce

3.1.6.1 Attributes

No additional attributes

3.1.6.2 Generalizations

None

3.1.6.3 Operations

No additional operations