

Table Of Content

vision	2
Config	2
Vision	3
vision.controller	4
Controller	4
HeaterController	6
PluginController	7
WindowController	9
vision.model	11
Database	12
Groundplan	14
JSONConverter	14
Model	16
ObjectFactory	20
PluginLoader	21
Position	21
Sample	23
Sensor	26
StaticObject	30
Update	30
UpdateThread	32
Wall	33
vision.view	34
GuiAppState	34
HeaterPlugin	35
MainAppState	36
Plugin	37
View	40
WindowPlugin	42
Index	45

Package vision

Class Summary

[Config](#)

Holds global configuration variables

[Vision](#)

Main class starts up the whole software

vision

Class Config

```
java.lang.Object
|
+--vision.Config
```

< [Fields](#) > < [Constructors](#) >

```
public class Config
extends java.lang.Object
```

Holds global configuration variables

Fields

serverUrl

```
public static final java.lang.String serverUrl
    defines where to fetch the sensor updates
```

updateIntervall

```
public static final int updateIntervall
    defines how often the sensor data is refreshed
```

Constructors

Config

```
public Config()
```

vision

Class Vision

```
java.lang.Object
|
+--vision.Vision
```

< [Constructors](#) > < [Methods](#) >

```
public class Vision
extends java.lang.Object
```

Main class starts up the whole software

Constructors

Vision

```
public Vision()
```

Methods

main

```
public static void main(java.lang.String[] args)
```

Package vision.controller

Class Summary

[Controller](#)

the main controller that passes key presses, user inputs and events to the subcontrollers and the model

[HeaterController](#)

Controller for the heater-plugin

[PluginController](#)

abstract superclass for all plugin controllers

[WindowController](#)

vision.controller

Class Controller

```
java.lang.Object
|
+--vision.controller.Controller
```

< [Constructors](#) > < [Methods](#) >

```
public class Controller
extends java.lang.Object
```

the main controller that passes key presses, user inputs and events to the subcontrollers and the model

Constructors

Controller

```
public Controller()
```

Methods

bind

```
public void bind(Nifty arg0,
                 Screen arg1)
```

binds the nifty instance to this controller

getModel

```
public Model getModel()
```

Getter of the property model

Returns:

Returns the model.

getPluginController

```
public java.util.Collection getPluginController()
```

Getter of the property pluginController

Returns:

Returns the pluginController.

getView

```
public View getView()
```

Getter of the property view

Returns:

Returns the view.

onEndScreen

```
public void onEndScreen()
```

onStartScreen

```
public void onStartScreen()
```

pluginButton

```
public void pluginButton(java.lang.String id)
```

pluginButton gets called by nifty if a button of a plugin was pressed and forwards it to the respective plugin controller

pluginCheckbox

```
public void pluginCheckbox(java.lang.String id)
```

gets called by nifty if a checkbox of a plugin was pressed and forwards it to the respective plugin controller

setModel

```
public void setModel(Model model)
```

Setter of the property model

Parameters:

model - The model to set.

setPluginController

```
public void setPluginController(java.util.Collection pluginController)
```

Setter of the property pluginController

Parameters:

pluginController - The pluginController to set.

setView

```
public void setView(View view)
```

Setter of the property view

Parameters:

view - The view to set.

vision.controller

Class HeaterController

```
java.lang.Object
|
+--PluginController
    |
    +--vision.controller.HeaterController
```

```
public class HeaterController
extends PluginController
```

Controller for the heater-plugin

Constructors

HeaterController

```
public HeaterController()
```

Methods

getHeaterPlugin

```
public HeaterPlugin getHeaterPlugin()
```

Getter of the property heaterPlugin

Returns:

Returns the heaterPlugin.

setHeaterPlugin

```
public void setHeaterPlugin(HeaterPlugin heaterPlugin)
```

Setter of the property heaterPlugin

Parameters:

heaterPlugin - The heaterPlugin to set.

vision.controller

Class PluginController

```
java.lang.Object
|
+--vision.controller.PluginController
```

Direct Known Subclasses:

[HeaterController](#), [WindowController](#)

< [Constructors](#) > < [Methods](#) >

```
public abstract class PluginController
extends java.lang.Object
```

abstract superclass for all plugin controllers

Constructors

PluginController

```
public PluginController(Model model,  
                       Plugin plugin)
```

constructs a new PluginController

Parameters:

model - the model

plugin - the plugin to manage

Methods

buttonPressed

```
public void buttonPressed(java.lang.String id)
```

callback function that gets called by the main controller if the user clicks on a plugin buttons

Parameters:

id - the id of the button

createButtons

```
public java.util.Map createButtons()
```

returns a List of plugin-defined buttons that the main system creates for the plugin

Returns:

a Map of button ids and their Text

createCheckBoxes

```
public java.util.Map createCheckBoxes()
```

returns a List of plugin-defined checkboxes (options) that the main system creates for the plugin

Returns:

a Map of checkbox ids and their texts

getModel

```
public Model getModel()
```

Getter of the property model

Returns:

Returns the model.

setModel

```
public void setModel(Model model)
```

Setter of the property model

Parameters:

model - The model to set.

vision.controller

Class WindowController

```
java.lang.Object
|
+--PluginController
    |
    +--vision.controller.WindowController
```

< [Constructors](#) > < [Methods](#) >

```
public class WindowController
extends PluginController
```

Constructors

WindowController

```
public WindowController()
```

Methods

getWindowPlugin

```
public WindowPlugin getWindowPlugin()
```

Getter of the property windowPlugin

Returns:

Returns the windowPlugin.

setWindowPlugin

```
public void setWindowPlugin(WindowPlugin windowPlugin)
```

Setter of the property windowPlugin

Parameters:

windowPlugin - The windowPlugin to set.

Package vision.model

Class Summary

[Database](#)

manages the database connection and saves sensordata

[Groundplan](#)

contains all static models, building architecture.

[JSONConverter](#)

[Model](#)

provides a facade for all objects belonging to the model

[ObjectFactory](#)

stub - autogenerated by jaxb

[PluginLoader](#)

loads all plugins from a configured subdirectory

[Position](#)

3-dimensional vector

[Sample](#)

holds a sensor measurement and the time it was taken

[Sensor](#)

The Sensor class holds all sensor data

[StaticObject](#)

implements a static object in the environment

[Update](#)

manages the server connection and fetches the sensor data

[UpdateThread](#)

updates the sensor data in the background.

[Wall](#)

vision.model

Class Database

```
java.lang.Object
|
+--vision.model.Database
```

< [Constructors](#) > < [Methods](#) >

```
public class Database
extends java.lang.Object
```

manages the database connection and saves sensordata

Constructors

Database

```
public Database()
```

Methods

getAllSensorData

```
public java.util.List getAllSensorData(int id)
```

Parameters:

id - id of the sensor

Returns:

a list of all sensor samples belonging to the given sensor

getDaten

```
public Model getDaten()
```

Getter of the property daten

Returns:

Returns the daten.

getSensordata

```
public java.util.List getSensordata(int id,  
                                     int zeitpunkt)
```

fetches the sensor samples collected

getUpdate

```
public Update getUpdate()
```

Getter of the property update

Returns:

Returns the update.

setDaten

```
public void setDaten(Model daten)
```

Setter of the property daten

Parameters:

daten - The daten to set.

setUpdate

```
public void setUpdate(Update update)
```

Setter of the property update

Parameters:

update - The update to set.

updateSensors

```
public void updateSensors(java.lang.String id,  
                           int zeitpunkt,  
                           Sample messwerte)
```

saves a sensor object in the database

vision.model

Class Groundplan

```
java.lang.Object
|
+--vision.model.Groundplan
```

< [Constructors](#) > < [Methods](#) >

```
public class Groundplan
extends java.lang.Object
```

contains all static models, building architecture.

Constructors

Groundplan

```
public Groundplan()
```

Methods

load

```
public void load()
```

vision.model

Class JSONConverter

```
java.lang.Object
|
+--vision.model.JSONConverter
```

< [Constructors](#) > < [Methods](#) >

```
public class JSONConverter
extends java.lang.Object
```

Constructors

JSONConverter

```
public JSONConverter()
```

Methods

addSensorToList

```
public void addSensorToList()
```

convert

```
public void convert()
```

getSensorList

```
public java.util.List getSensorList()
```

getUpdate

```
public Update getUpdate()
```

Getter of the property update

Returns:

Returns the update.

getUrl

```
public void getUrl()
```

setUpdate

```
public void setUpdate(Update update)
```

Setter of the property update

Parameters:

update - The update to set.

vision.model

Class Model

```
java.lang.Object
|
+--vision.model.Model
```

< [Constructors](#) > < [Methods](#) >

```
public class Model
extends java.lang.Object
```

provides a facade for all objects belonging to the model

Constructors

Model

```
public Model(View view)
```

Methods

getDatenbank

```
public Database getDatenbank()
```

Getter of the property datenbank

Returns:

Returns the datenbank.

getGroundplan

```
public Groundplan getGroundplan()
```

Getter of the property groundplan

Returns:

Returns the groundplan.

getPlugin

```
public java.util.Collection getPlugin()
```

Getter of the property plugin

Returns:

Returns the plugin.

getPluginList

```
public java.util.List getPluginList()
```

Getter of the property pluginList

Returns:

Returns the pluginList.

getPluginLoader

```
public PluginLoader getPluginLoader()
```

Getter of the property pluginLoader

Returns:

Returns the pluginLoader.

getSensor

```
public java.util.Collection getSensor()
```

Getter of the property sensor

Returns:

Returns the sensor.

getSensordata

```
public void getSensordata(java.lang.String id,  
                           int time)
```

getTaggedSensors

```
public void getTaggedSensors(java.util.List tags)
```

getUpdate

```
public Update getUpdate()
```

Getter of the property update

Returns:

Returns the update.

getView

```
public View getView()
```

Getter of the property view

Returns:

Returns the view.

setDatenbank

```
public void setDatenbank(Database datenbank)
```

Setter of the property datenbank

Parameters:

datenbank - The datenbank to set.

setGroundplan

```
public void setGroundplan(Groundplan groundplan)
```

Setter of the property groundplan

Parameters:

groundplan - The groundplan to set.

setPlugin

```
public void setPlugin(java.util.Collection plugin)
```

Setter of the property plugin

Parameters:

plugin - The plugin to set.

setPluginList

```
public void setPluginList(java.util.List pluginList)
```

Setter of the property pluginList

Parameters:

pluginList - The pluginList to set.

setPluginLoader

```
public void setPluginLoader(PluginLoader pluginLoader)
```

Setter of the property pluginLoader

Parameters:

pluginLoader - The pluginLoader to set.

setSensor

```
public void setSensor(java.util.Collection sensor)
```

Setter of the property sensor

Parameters:

sensor - The sensor to set.

setUpdate

```
public void setUpdate(Update update)
```

Setter of the property update

Parameters:

update - The update to set.

setView

```
public void setView(View view)
```

Setter of the property view

Parameters:

view - The view to set.

vision.model

Class ObjectFactory

```
java.lang.Object
|
+--vision.model.ObjectFactory
```

< [Constructors](#) >

class **ObjectFactory**
extends java.lang.Object

stub - autogenerated by jaxb

Constructors

ObjectFactory

```
ObjectFactory( )
```

vision.model

Class PluginLoader

```
java.lang.Object
|
+--vision.model.PluginLoader
```

< [Constructors](#) > < [Methods](#) >

```
public class PluginLoader
extends java.lang.Object
```

loads all plugins from a configured subdirectory

Constructors

PluginLoader

```
public PluginLoader()
```

Methods

loadPlugins

```
public java.util.List loadPlugins()
```

vision.model

Class Position

```
java.lang.Object
|
+--vision.model.Position
```

< [Constructors](#) > < [Methods](#) >

```
public class Position
extends java.lang.Object
```

3-dimensional vector

Constructors

Position

```
public Position()
```

Methods

getSensor

```
public Sensor getSensor()
```

Getter of the property sensor

Returns:

Returns the sensor.

getX

```
public float getX()
```

Getter of the property x

Returns:

Returns the x.

getY

```
public float getY()
```

Getter of the property y

Returns:

Returns the y.

getZ

```
public float getZ()
```

Getter of the property z

Returns:

Returns the z.

setSensor

```
public void setSensor(Sensor sensor)
```

Setter of the property sensor

Parameters:

sensor - The sensor to set.

setX

```
public void setX(float x)
```

Setter of the property x

Parameters:

x - The x to set.

setY

```
public void setY(float y)
```

Setter of the property y

Parameters:

y - The y to set.

setZ

```
public void setZ(float z)
```

Setter of the property z

Parameters:

z - The z to set.

vision.model

Class Sample

```
java.lang.Object  
|  
+--vision.model.Sample
```

< [Constructors](#) > < [Methods](#) >

```
public class Sample  
extends java.lang.Object
```

holds a sensor measurement and the time it was taken

Constructors

Sample

```
public Sample()
```

Methods

getMesswert

```
public void getMesswert()
```

getSensor

```
public Sensor getSensor()
```

Getter of the property sensor

Returns:

Returns the sensor.

getTyp

```
public java.lang.String getTyp()
```

Getter of the property typ

Returns:

Returns the typ.

getUnit

```
public java.lang.String getUnit()
```

Getter of the property unit

Returns:

Returns the unit.

getUpdate

```
public int getUpdate()
```

Getter of the property update

Returns:

Returns the update.

getValue

```
public float getValue()
```

Getter of the property value

Returns:

Returns the value.

setMesswert

```
public void setMesswert()
```

setSensor

```
public void setSensor(Sensor sensor)
```

Setter of the property sensor

Parameters:

sensor - The sensor to set.

setTyp

```
public void setTyp(java.lang.String typ)
```

Setter of the property typ

Parameters:

typ - The typ to set.

setUnit

```
public void setUnit(java.lang.String unit)
```

Setter of the property unit

Parameters:

unit - The unit to set.

setUpdate

```
public void setUpdate(int update)
```

Setter of the property update

Parameters:

update - The update to set.

setValue

```
public void setValue(float value)
```

Setter of the property value

Parameters:

value - The value to set.

vision.model

Class Sensor

```
java.lang.Object  
|  
+--vision.model.Sensor
```

< [Constructors](#) > < [Methods](#) >

```
public class Sensor  
extends java.lang.Object
```

The Sensor class holds all sensor data

Constructors

Sensor

```
public Sensor()
```

Methods

getDescription

```
public java.lang.String getDescription()
```

Getter of the property Description

Returns:

Returns the description.

getId

```
public java.lang.String getId()
```

Getter of the property id

Returns:

Returns the id.

getMesswert

```
public Sample getMesswert()
```

Getter of the property messwert

Returns:

Returns the messwert.

getPosition

```
public Position getPosition()
```

Getter of the property position

Returns:

Returns the position.

getTags

```
public java.util.List getTags()
```

Getter of the property tags

Returns:

Returns the tags.

getUpdate

```
public int getUpdate()
```

Getter of the property update

Returns:

Returns the update.

isRegistered

```
public boolean isRegistered()
```

Getter of the property registered

Returns:

Returns the registered.

setDescription

```
public void setDescription(java.lang.String description)
```

Setter of the property Description

Parameters:

Description - The description to set.

setId

```
public void setId(java.lang.String id)
```

Setter of the property id

Parameters:

id - The id to set.

setMesswert

```
public void setMesswert(Sample messwert)
```

Setter of the property messwert

Parameters:

messwert - The messwert to set.

setPosition

```
public void setPosition(Position position)
```

Setter of the property position

Parameters:

position - The position to set.

setRegistered

```
public void setRegistered(boolean registered)
```

Setter of the property registered

Parameters:

registered - The registered to set.

setTags

```
public void setTags(java.util.List tags)
```

Setter of the property tags

Parameters:

tags - The tags to set.

setUpdate

```
public void setUpdate(int update)
```

Setter of the property update

Parameters:

update - The update to set.

vision.model

Class StaticObject

```
java.lang.Object
|
+--vision.model.StaticObject
```

< [Constructors](#) >

```
public class StaticObject
extends java.lang.Object
```

implements a static object in the environment

Constructors

StaticObject

```
public  StaticObject()
```

vision.model

Class Update

```
java.lang.Object
|
+--vision.model.Update
```

< [Constructors](#) > < [Methods](#) >

```
public class Update
extends java.lang.Object
```

manages the server connection and fetches the sensor data

Constructors

Update

```
public  Update()
```

Methods

getAllData

```
public void getAllData()
```

getDatabase

```
public Database getDatabase()
```

Getter of the property database

Returns:

Returns the database.

getDaten

```
public Model getDaten()
```

Getter of the property daten

Returns:

Returns the daten.

getJSONConverter

```
public JSONConverter getJSONConverter()
```

Getter of the property jsonConverter

Returns:

Returns the jsonConverter.

setDatabase

```
public void setDatabase(Database database)
```

Setter of the property database

Parameters:

database - The database to set.

setDaten

```
public void setDaten(Model daten)
```

Setter of the property daten

Parameters:

daten - The daten to set.

setJSONConverter

```
public void setJSONConverter(JSONConverter jsonConverter)
```

Setter of the property jsonConverter

Parameters:

jsonConverter - The jsonConverter to set.

store

```
public void store(int time)
```

vision.model

Class UpdateThread

```
java.lang.Object
|
+-- java.lang.Thread
|
+-- vision.model.UpdateThread
```

All Implemented Interfaces:

java.lang.Runnable

< [Constructors](#) > < [Methods](#) >

```
public class UpdateThread
extends java.lang.Thread
```

updates the sensor data in the background.

Constructors

UpdateThread

```
public UpdateThread()
```

Methods

getUpdate

```
public Update getUpdate()
```

Getter of the property update

Returns:

Returns the update.

setUpdate

```
public void setUpdate(Update update)
```

Setter of the property update

Parameters:

update - The update to set.

vision.model

Class Wall

```
java.lang.Object
|
+--vision.model.Wall
```

< [Constructors](#) >

```
public class Wall
extends java.lang.Object
```

Constructors

Wall

```
public Wall()
```

Package vision.view

Class Summary

[GuiAppState](#)

renders the user interface

[HeaterPlugin](#)

[MainAppState](#)

Renders all static objects and rooms

[Plugin](#)

[View](#)

main class of the view package.

[WindowPlugin](#)

vision.view

Class GuiAppState

```
java.lang.Object
|
+--AbstractAppState
|
+--vision.view.GuiAppState
```

< [Constructors](#) > < [Methods](#) >

```
public class GuiAppState
extends AbstractAppState
```

renders the user interface

Constructors

GuiAppState

```
public GuiAppState()
```

Methods

initialize

```
public void initialize(AppStateManager stateManager,  
                        Application app)
```

vision.view

Class HeaterPlugin

```
java.lang.Object  
|  
+--AbstractAppState  
|  
+--Plugin  
|  
+--vision.view.HeaterPlugin
```

< [Constructors](#) > < [Methods](#) >

```
public class HeaterPlugin  
extends Plugin
```

Author:

idle This class represents the plugins of the heater

Constructors

HeaterPlugin

```
public HeaterPlugin()
```

Methods

clientUpdate

```
protected void clientUpdate(Application application)
```

updates the client

Overrides:

[clientUpdate](#) in class [Plugin](#)

getHeaterController

```
public HeaterController getHeaterController()
```

Getter of the property heaterController

Returns:

Returns the heaterController.

getHeaters

```
public java.util.List getHeaters()
```

Getter of the property heaters

Returns:

Returns the heaters1.

setHeaterController

```
public void setHeaterController(HeaterController heaterController)
```

Setter of the property heaterController

Parameters:

heaterController - The heaterController to set.

setHeaters

```
public void setHeaters(java.util.List heaters)
```

Setter of the property Heizungen

Parameters:

Heizungen - The heizungen to set.

vision.view

Class MainAppState

```
java.lang.Object
|
+--AbstractAppState
|
+--vision.view.MainAppState
```

```
public class MainAppState
extends AbstractAppState
```

Renders all static objects and rooms

Constructors

MainAppState

```
public MainAppState()
```

Methods

initialize

```
public void initialize(AppStateManager stateManager,
                        Application app)
```

vision.view

Class Plugin

```
java.lang.Object
|
+--AbstractAppState
|
+--vision.view.Plugin
```

Direct Known Subclasses:

[HeaterPlugin](#), [WindowPlugin](#)

< [Constructors](#) > < [Methods](#) >

```
public abstract class Plugin
extends AbstractAppState
```

Constructors

Plugin

```
public Plugin()
```

Methods

clientUpdate

protected void **clientUpdate**(Application application)

getApp

public Application **getApp**()

Getter of the property app

Returns:

Returns the app.

getApplication

protected Application **getApplication**()

getDaten

public [Model](#) **getDaten**()

Getter of the property daten

Returns:

Returns the daten.

getSensors

public java.util.List **getSensors**()

Getter of the property sensors

Returns:

Returns the sensors.

getTags

```
public java.lang.String[] getTags()
```

Getter of the property tags

Returns:

Returns the tags.

initialize

```
public void initialize(AppStateManager stateManager,  
                        Application app)
```

setDaten

```
public void setDaten(Model daten)
```

Setter of the property daten

Parameters:

daten - The daten to set.

setSensors

```
public void setSensors(java.util.List sensors)
```

Setter of the property sensors

Parameters:

sensors - The sensors to set.

setTags

```
public void setTags(java.lang.String[] tags)
```

Setter of the property tags

Parameters:

tags - The tags to set.

update

```
public void update(Application application)
```

vision.view

Class View

```
java.lang.Object
|
+--SimpleApplication
|
+--vision.view.View
```

< [Constructors](#) > < [Methods](#) >

```
public class View
extends SimpleApplication
```

main class of the view package. contains the main update loop and calls the plugin and main views

Constructors

View

```
public View()
```

Methods

getController

```
public Controller getController()
```

Getter of the property controller

Returns:

Returns the controller.

getDaten

```
public Model getDaten()
```

Getter of the property daten

Returns:

Returns the daten.

getGuiAppState

```
public GuiAppState getGuiAppState()
```

Getter of the property guiAppState

Returns:

Returns the guiAppState.

getMainAppState

```
public MainAppState getMainAppState()
```

Getter of the property mainAppState

Returns:

Returns the mainAppState.

setController

```
public void setController(Controller controller)
```

Setter of the property controller

Parameters:

controller - The controller to set.

setDaten

```
public void setDaten(Model daten)
```

Setter of the property daten

Parameters:

daten - The daten to set.

setGuiAppState

```
public void setGuiAppState(GuiAppState guiAppState)
```

Setter of the property guiAppState

Parameters:

guiAppState - The guiAppState to set.

setMainAppState

```
public void setMainAppState(MainAppState mainAppState)
```

Setter of the property mainAppState

Parameters:

mainAppState - The mainAppState to set.

simpleInitApp

```
public void simpleInitApp()
```

initializes the view

simpleUpdate

```
public void simpleUpdate()
```

is called every frame by jmonkey

vision.view

Class WindowPlugin

```
java.lang.Object
|
+--AbstractAppState
|   |
|   +--Plugin
|       |
|       +--vision.view.WindowPlugin
```

< [Constructors](#) > < [Methods](#) >

```
public class WindowPlugin
extends Plugin
```

Constructors

WindowPlugin

```
public WindowPlugin()
```

Methods

clientUpdate

```
protected void clientUpdate(Application application)
```

Overrides:

[clientUpdate](#) in class [Plugin](#)

getWindowController

```
public WindowController getWindowController()
```

Getter of the property windowController

Returns:

Returns the windowController.

getWindows

```
public java.util.List getWindows()
```

Getter of the property windows

Returns:

Returns the windows.

setWindowController

```
public void setWindowController(WindowController windowController)
```

Setter of the property windowController

Parameters:

windowController - The windowController to set.

setWindows

```
public void setWindows(java.util.List windows)
```

Setter of the property windows

Parameters:

windows - The windows to set.

INDEX

A

[addSensorToList](#) ... 15

B

[bind](#) ... 4

[buttonPressed](#) ... 8

C

[clientUpdate](#) ... 35

[clientUpdate](#) ... 38

[clientUpdate](#) ... 43

[convert](#) ... 15

[createButtons](#) ... 8

[createCheckBoxes](#) ... 8

[Config](#) ... 2

[Config](#) ... 2

[Controller](#) ... 4

[Controller](#) ... 4

D

[Database](#) ... 12

[Database](#) ... 12

G

[getAllData](#) ... 31

[getAllSensorData](#) ... 12

[getApp](#) ... 38

[getApplication](#) ... 38

[getController](#) ... 40

[getDatabase](#) ... 31

[getDaten](#) ... 12

[getDaten](#) ... 31

[getDaten](#) ... 38

[getDaten](#) ... 41

[getDatenbank](#) ... 16

[getDescription](#) ... 27

[getGroundplan](#) ... 17

[getGuiAppState](#) ... 41

[getHeaterController](#) ... 36

[getHeaterPlugin](#) ... 7

[getHeaters](#) ... 36

[getId](#) ... 27

[getJSONConverter](#) ... 31

[getMainAppState](#) ... 41

[getMesswert](#) ... 24

[getMesswert](#) ... 27

[getModel](#) ... 5

[getModel](#) ... 9

[getPlugin](#) ... 17

[getPluginController](#) ... 5

[getPluginList](#) ... 17

[getPluginLoader](#) ... 17

[getPosition](#) ... 27

[getSensor](#) ... 17

[getSensor](#) ... 22

[getSensor](#) ... 24

[getSensordata](#) ... 13

[getSensordata](#) ... 18

[getSensorList](#) ... 15

[getSensors](#) ... 38

[getTaggedSensors](#) ... 18

[getTags](#) ... 28

[getTags](#) ... 39

[getTyp](#) ... 24

[getUnit](#) ... 24

[getUpdate](#) ... 13

[getUpdate](#) ... 15

[getUpdate](#) ... 18

[getUpdate](#) ... 25

[getUpdate](#) ... 28

[getUpdate](#) ... 33

[getUrl](#) ... 15

[getValue](#) ... 25

[getView](#) ... 5

[getView](#) ... 18

[getWindowController](#) ... 43

[getWindowPlugin](#) ... 10

[getWindows](#) ... 43

[getX](#) ... 22

[getY](#) ... 22

[getZ](#) ... 22

[Groundplan](#) ... 14

[Groundplan](#) ... 14

[GuiAppState](#) ... 34

[GuiAppState](#) ... 34

H

[HeaterController](#) ... 6
[HeaterController](#) ... 7
[HeaterPlugin](#) ... 35
[HeaterPlugin](#) ... 35

I

[initialize](#) ... 35
[initialize](#) ... 37
[initialize](#) ... 39
[isRegistered](#) ... 28

J

[JSONConverter](#) ... 14
[JSONConverter](#) ... 15

L

[load](#) ... 14
[loadPlugins](#) ... 21

M

[main](#) ... 3
[MainAppState](#) ... 36
[MainAppState](#) ... 37
[Model](#) ... 16
[Model](#) ... 16

O

[onEndScreen](#) ... 5
[onStartScreen](#) ... 5
[ObjectFactory](#) ... 20
[ObjectFactory](#) ... 20

P

[pluginButton](#) ... 5
[pluginCheckbox](#) ... 6
[Plugin](#) ... 37
[Plugin](#) ... 37
[PluginController](#) ... 7
[PluginController](#) ... 8
[PluginLoader](#) ... 21
[PluginLoader](#) ... 21
[Position](#) ... 21
[Position](#) ... 22

S

[serverUrl](#) ... 2
[setController](#) ... 41
[setDatabase](#) ... 31
[setDaten](#) ... 13
[setDaten](#) ... 32
[setDaten](#) ... 39
[setDaten](#) ... 41
[setDatenbank](#) ... 18
[setDescription](#) ... 28
[setGroundplan](#) ... 19
[setGuiAppState](#) ... 42
[setHeaterController](#) ... 36
[setHeaterPlugin](#) ... 7
[setHeaters](#) ... 36
[setId](#) ... 28
[setJSONConverter](#) ... 32
[setMainAppState](#) ... 42
[setMesswert](#) ... 25
[setMesswert](#) ... 29
[setModel](#) ... 6
[setModel](#) ... 9
[setPlugin](#) ... 19
[setPluginController](#) ... 6
[setPluginList](#) ... 19
[setPluginLoader](#) ... 19
[setPosition](#) ... 29
[setRegistered](#) ... 29
[setSensor](#) ... 19
[setSensor](#) ... 23
[setSensor](#) ... 25
[setSensors](#) ... 39
[setTags](#) ... 29
[setTags](#) ... 39
[setTyp](#) ... 25
[setUnit](#) ... 26
[setUpdate](#) ... 13
[setUpdate](#) ... 16
[setUpdate](#) ... 20
[setUpdate](#) ... 26
[setUpdate](#) ... 29
[setUpdate](#) ... 33
[setValue](#) ... 26
[setView](#) ... 6
[setView](#) ... 20
[setWindowController](#) ... 43
[setWindowPlugin](#) ... 10
[setWindows](#) ... 44
[setX](#) ... 23
[setY](#) ... 23
[setZ](#) ... 23
[simpleInitApp](#) ... 42
[simpleUpdate](#) ... 42
[store](#) ... 32
[Sample](#) ... 23
[Sample](#) ... 24
[Sensor](#) ... 26
[Sensor](#) ... 27
[StaticObject](#) ... 30
[StaticObject](#) ... 30

U

[update](#) ... 40
[updateIntervall](#) ... 2
[updateSensors](#) ... 13
[Update](#) ... 30
[Update](#) ... 30
[UpdateThread](#) ... 32
[UpdateThread](#) ... 33

V

[View](#) ... 40
[View](#) ... 40
[Vision](#) ... 3
[Vision](#) ... 3

W

[Wall](#) ... 33
[Wall](#) ... 33
[WindowController](#) ... 9
[WindowController](#) ... 9
[WindowPlugin](#) ... 42
[WindowPlugin](#) ... 43