## VARIABLES

- backgroundcolor (FFFFFF)
- windowless (false)
- height (260)
- width (320)

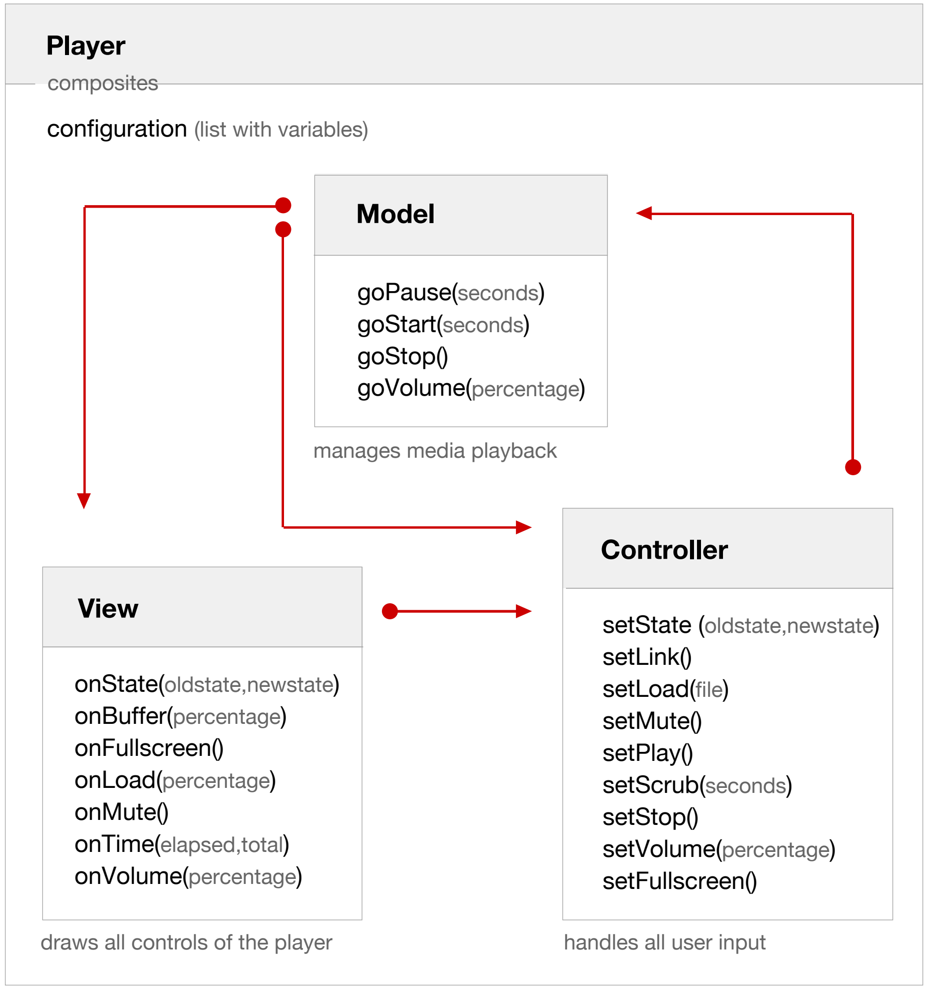
- duration (0)
- file (null)
- image (null)
- link (null)
- start (0)

- backcolor (FFFFFF)
- frontcolor (000000)
- lightcolor (000000)
- screencolor (000000)

- logo (null)
- overstretch (false)
- shownavigation (true)
- showstop (false)
- showdigits (true)

- autostart (false)
- bufferlength (3)
- linkfromdisplay (false)
- linktarget (_self)
- repeat (false)
- sender (null)
- usefullscreen (true)
- usemute (false)
- volume (90)

## STRUCTURE



When the Player starts, it first loads the configuration list and places the graphics on the page. Next, Controller, Model and View are created. They now start calling each others' functions in a so-called MVC loop. Model calls the functions from View, View those from Controller and Controller those from Model. There is one exception: Model calls both the onState() from View and the setState() from Controller.

## JAVASCRIPT API

Since Silverlight programming is done in javascript, the player is easily accessible. Javascript has already got a reference to the player through the instantiation:

```
var ply = new jeroenwijering.Player(cnt,src,cfg);
```

Requesting a configuration variable can be done through this reference:

```
alert('the file is: '+ply.getConfig().file);
```

The player can be controlled by sending events. Each function of Controller is named after these events. Examples:

```
ply.sendEvent('PLAY');
ply.sendEvent('VOLUME',50);
ply.sendEvent('LOAD','newvideo.wmv');
```

Finally, you can assign your own functions to listen to specific events of the player. Each function of View is named after the corresponding event:

```
ply.addListener('VOLUME',alert);
function myFunc(old,new) {
  alert('the state went from '+old+' to '+new);
};
ply.addListener('STATE',myFunc);
```

Note that the state event can have the values Buffering, Closed, Error, Opening, Paused, Playing, Stopped, or Completed.