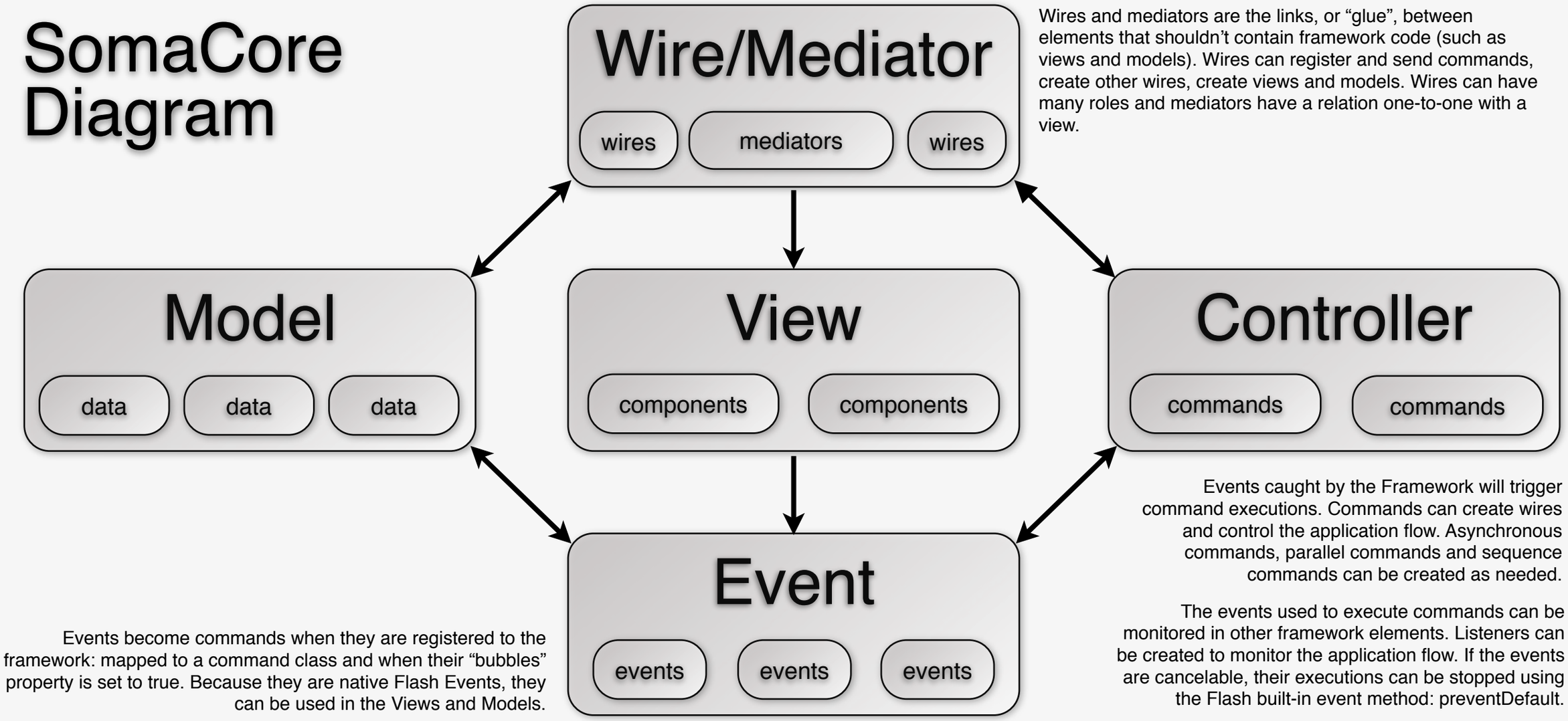
SomaCore Diagram

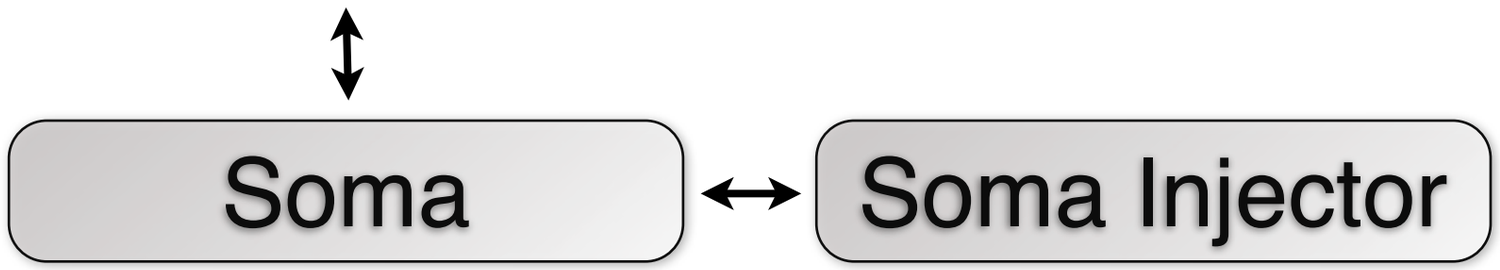


Wires and mediators are the links, or “glue”, between elements that shouldn’t contain framework code (such as views and models). Wires can register and send commands, create other wires, create views and models. Wires can have many roles and mediators have a relation one-to-one with a view.

Events become commands when they are registered to the framework: mapped to a command class and when their “bubbles” property is set to true. Because they are native Flash Events, they can be used in the Views and Models.

Events caught by the Framework will trigger command executions. Commands can create wires and control the application flow. Asynchronous commands, parallel commands and sequence commands can be created as needed.

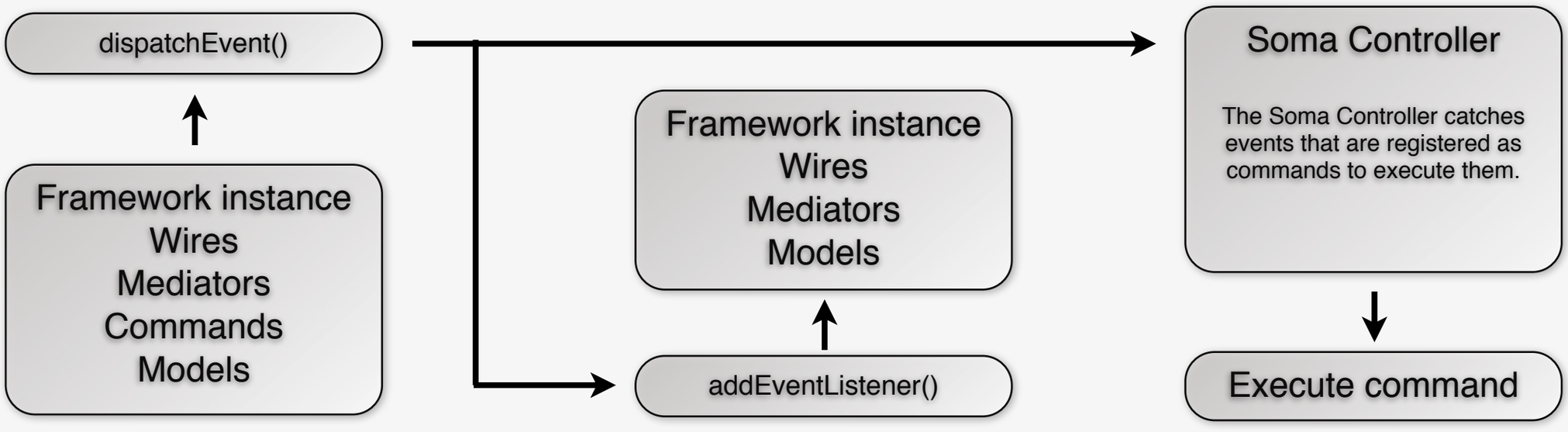
The events used to execute commands can be monitored in other framework elements. Listeners can be created to monitor the application flow. If the events are cancelable, their executions can be stopped using the Flash built-in event method: preventDefault.



There is no Singleton in Soma,
You can extend the class Soma (and implement ISoma) to start to build your own application.
Your Soma Application will be the gate to talk to the Framework and is easily accessed from the Wires, the Mediators and the Commands. Your application can create Wires and Mediators, register Commands, instantiates Soma Plugins, initialise the application, and so on.

SomaCore

Commands flow from framework elements



Wires, Mediators, Commands or Models are using the same EventDispatcher instance: the Framework instance itself.

Commands can be dispatched from all these Framework elements using the Flash built-in method:

dispatchEvent(MyEvent.DO_SOMETHING);

Important note: the events dispatched must have their property "bubbles" set to true. Otherwise, the framework will ignored the events.

The Framework instance, Wires, Mediators and Models can listen to commands to monitor the application flow using Flash built-in methods:

addEventListener(MyEvent.DO_SOMETHING, handler);

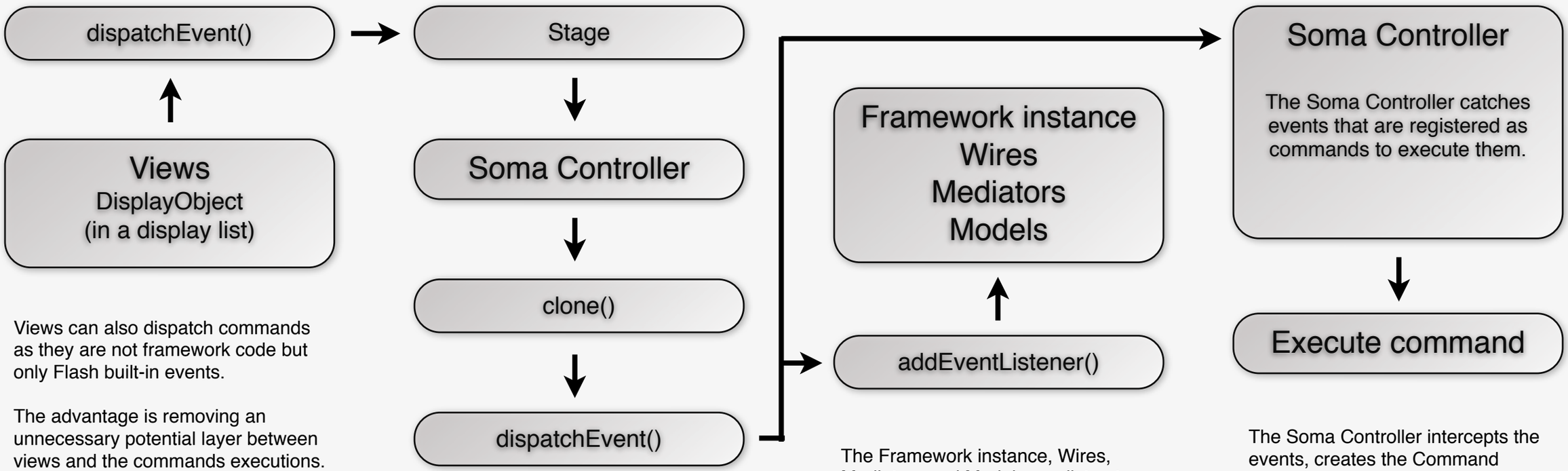
If the events are cancelable, the execution of the commands can be stopped before they occurred using the Flash built-in method:

event.preventDefault();

The Soma Controller intercepts the command dispatched through the Framework, creates the Command instance and calls its execute method.

The execution of the command will not occur if the event has been previously "default prevented".

SomaCore
Commands flow from views



Views can also dispatch commands as they are not framework code but only Flash built-in events.

The advantage is removing an unnecessary potential layer between views and the commands executions.

The only requirement is that the view must be on the stage (or any display list) when the command is dispatched as the framework is using the stage to capture events from views.

The commands can be dispatched from the views using the Flash built-in method:

dispatchEvent
(MyEvent.DO_SOMETHING);

Important note: the events dispatched must have their property "bubbles" set to true. Otherwise, the framework will ignored the events.

The Soma Controller intercepts each command, makes a clone of the event and dispatches the clone for application flow monitoring.

These actions are transparent and automatically done by the framework.

Important note: not only a good practice, it is also important that the custom events override the clone method not to lose any custom event properties.

The Framework instance, Wires, Mediators and Models can listen to commands to monitor the application flow using Flash built-in methods:

addEventListener
(MyEvent.DO_SOMETHING, handler);

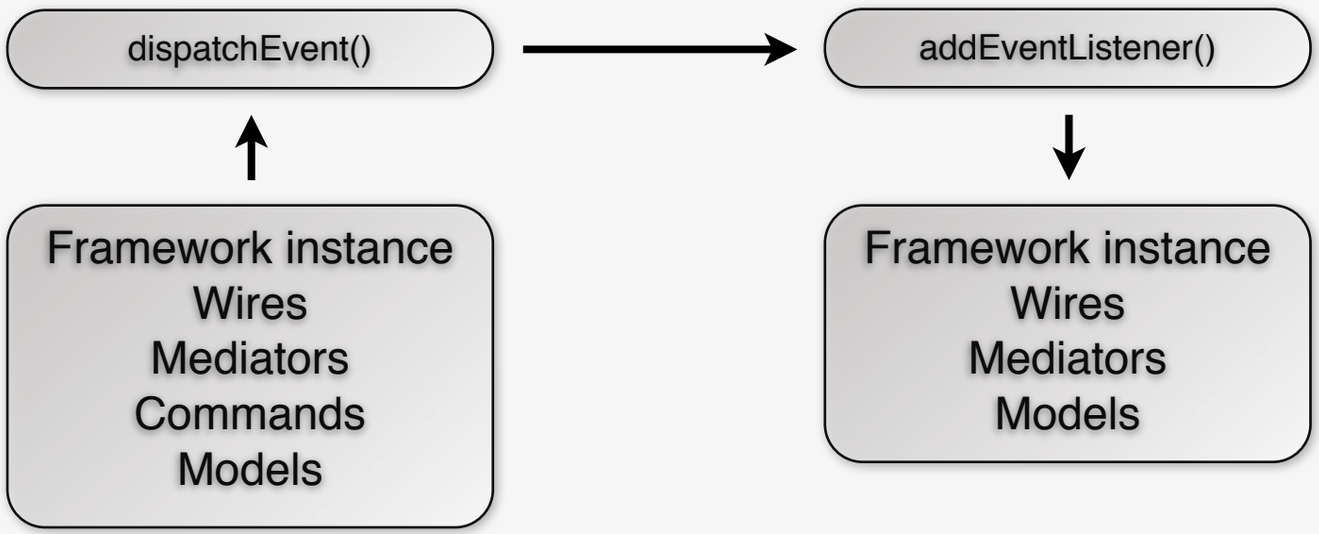
If the events are cancelable, the execution of the commands can be stopped before they occurred using the Flash built-in method:

event.preventDefault();

The Soma Controller intercepts the events, creates the Command instance and calls its execute method.

The execution of the command will not occur if the event has been previously "default prevented".

SomaCore
Events flow from framework elements



Wires, Mediators, Commands or Models are using the same EventDispatcher instance: the framework instance itself.

Events can be dispatched, without being registered as commands to the framework.

```
dispatchEvent(MyEvent.DO_SOMETHING);
```

Others framework elements such as Wires, Mediators and Models can listen to simple events (i.e. events that are not commands) dispatched from other framework elements.

```
addEventListener(MyEvent.DO_SOMETHING, handler);
```