Model-View-Controller (MVC) Pattern

- An *architectural pattern* used to help separate the external *view* of an application from the logic
  - The ***model*** contains the underlying classes containing the logic and state of the application
  - The ***view*** contains objects used to render the appearance of the data from the model in the user interface
  - The ***controller*** handles the user’s interaction with the view and the model

MVC Applications

- Practically all web frameworks use MVC.

- Angular.js - forces you to use MVC on the client side (in Javascript). Client-server design is yet another MVC.

- GUI Interfaces, like *Swing* (see *Core Java,* p.339-345)

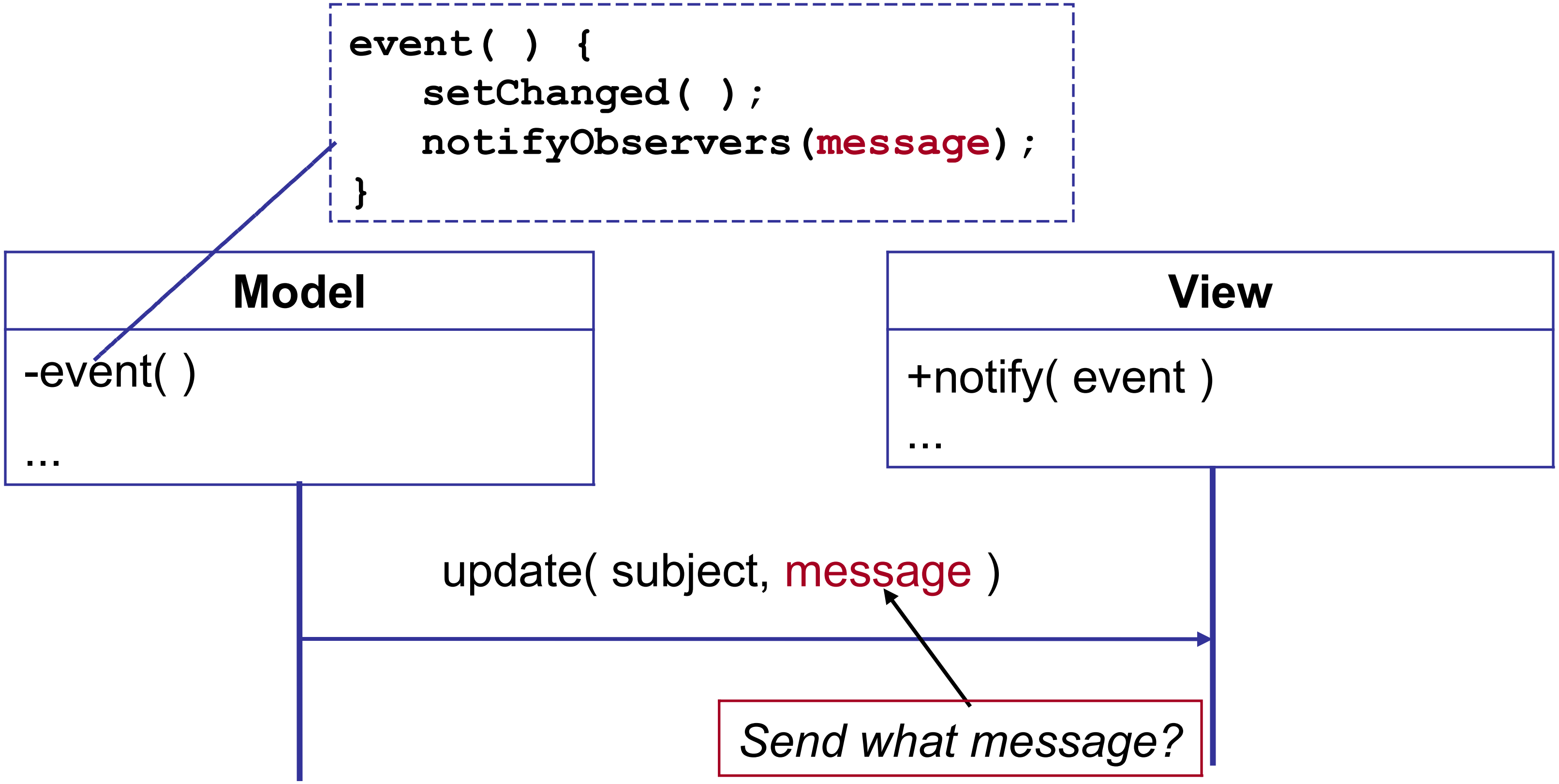
- Editors: *Dreamweaver*

Model and Views

The model and **view** relationship is often implemented as the Observable Pattern.

This lets the application have multiple views.

Reduces "coupling" between model and view.



```
event( ) {
    setChanged( );
    notifyObservers(message);
}
```

Model

-event( )

...

View

+notify(event)

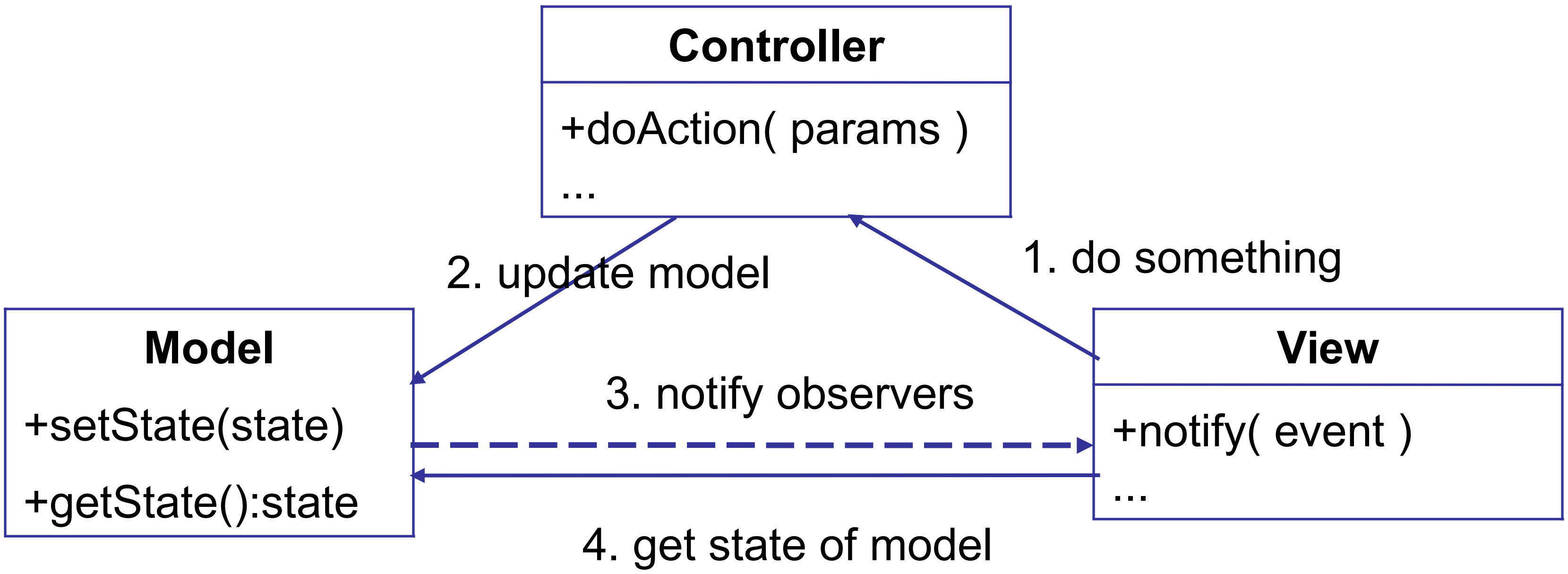
...

update(subject, message)

Send what message?

View and Controller

The view calls the controller class when it wants to do something.


Controller
+doAction(params)
...
2. update model
1. do something
Model
+setState(state)
+getState():state
View
3. notify observers
+notify(event)
...
4. get state of model