

```

Loading vc-cvs...
1 package caltool.view;
2
3 import caltool.schedule.*;
4 import caltool.caldb.*;
5 import mvp.*;
6
7 /**
8  *
9  * Class View is the top-level model class in the view package. It provides
10 * methods to view the calendar at the five structural levels of a calendar:
11 * item, day, week, month, and year. There are also methods to go to the
12 * previous and next views at any level, as well as an method to go to a
13 * specific date. Methods are provided to view lists of scheduled items in a
14 * variety of ways. Methods are provided to view other users' calendars and to
15 * view a list of active viewing windows. View filtering are capabilities are
16 * defined in the Filter submodel.
17 *
18 */
19 public class View extends Model {
20
21     public View(mvp.View view, CalendarDB caldb) {
22         super(view);
23
24         this.caldb = caldb;
25
26         dailyAgenda = new DailyAgenda(caldb);
27         weeklyAgendaTable = new WeeklyAgendaTable(caldb);
28         weeklyAgendaList = new WeeklyAgendaList(caldb);
29         monthlyAgenda = new MonthlyAgenda(caldb);
30         lists = new Lists(caldb);
31
32         appointmentsHidden = false;
33     }
34
35     /*-
36     * Derived methods
37     */
38
39     /**
40     * Produce the currently selected scheduled item.
41     */
42     public ScheduledItem viewItem() {
43         return caldb.getCurrentCalendar().getSelectedItem();
44     }
45
46     /**
47     * Produce the daily agenda for the currently selected date, or for today's
48     * date if no other date is currently selected.
49     */
50     public DailyAgenda viewDay() {
51         dailyAgenda.update(null, null);
52         return null;
53     }
54
55     /**
56
57     * Produce the monthly agenda for the currently selected date, or for
58     * today's date if no other date is currently selected.
59     */
60     public MonthlyAgenda viewMonth() {
61         monthlyAgenda.update(null, null);
62         return monthlyAgenda;
63     }
64
65     /**
66     * Return the lists model class that has the methods to compute the differe
67     * forms of lists.
68     */
69     public Lists getLists() {
70         return lists;
71     }
72
73     /**
74     * Select the date in the current calendar. The most typical reason for
75     * date selection is as the argument to a view command.
76     */
77     public void selectDate(Date date) {
78         calDB.setSelectedDate(date);
79     }
80
81     /**
82     * Select the date given a single date number. Figure out the complete
83     * date based on the currently active view window.
84     */
85     public void selectDate(int date) {
86         System.out.println(date);
87     }
88
89     /**
90     * Toggle the show/hide state for appointments.
91     */
92     public void toggleShowHideAppointments() {
93         appointmentsHidden = ! appointmentsHidden;
94     }
95
96     /**
97     * Return true if appointments are hidden, false if not.
98     */
99     public boolean areAppointmentsHidden() {
100         return appointmentsHidden;
101     }
102
103     /**
104     * Toggle the show/hide state for meetings.
105     */
106     public void toggleShowHideMeetings() {
107         meetingsHidden = ! meetingsHidden;
108     }
109
110     /**
111     * Return true if meetings are hidden, false if not.

```

```

112     */
113     public boolean areMeetingsHidden() {
114         return meetingsHidden;
115     }
116
117     /**
118     * Toggle the show/hide state for tasks.
119     */
120     public void toggleShowHideTasks() {
121         tasksHidden = ! tasksHidden;
122     }
123
124     /**
125     * Return true if tasks are hidden, false if not.
126     */
127     public boolean areTasksHidden() {
128         return tasksHidden;
129     }
130
131     /**
132     * Toggle the show/hide state for events.
133     */
134     public void toggleShowHideEvents() {
135         eventsHidden = ! eventsHidden;
136     }
137
138     /**
139     * Return true if events are hidden, false if not.
140     */
141     public boolean areEventsHidden() {
142         return eventsHidden;
143     }
144
145     /** Calendar database in which viewed items are stored */
146     protected CalendarDB calDB;
147
148     /** The current instance of DailyAgenda that computes the view for the
149     * user-selected day. This must be refined into a collection of some form
150     * to support multi-window mode. */
151     protected DailyAgenda dailyAgenda;
152
153     /** The current instance of WeeklyAgendaTable that computes the view for
154     * the user-selected week. This must be refined into an collection of some
155     * form to support multi-window mode. */
156     protected WeeklyAgendaTable weeklyAgendaTable;
157
158     /** The current instance of WeeklyAgendaList that computes the view for the
159     * user-selected week. This must be refined into a collection of some form
160     * to support multi-window mode. */
161     protected WeeklyAgendaList weeklyAgendaList;
162
163     /** The current instance of MonthlyAgenda that computes the view for the
164     * user-selected month. This must be refined into a collection of some
165     * form to support multi-window mode. */
166     protected MonthlyAgenda monthlyAgenda;
167
168     /** The lists submodel */
169     protected Lists lists;
170
171     /** The filter submodel */
172     protected Filter filter;
173
174     /** The windows submodel */
175     protected Windows windows;
176
177     /** The shown/hidden state of appointments */
178     boolean appointmentsHidden;
179
180     /** The shown/hidden state of meetings */
181     boolean meetingsHidden;
182
183     /** The shown/hidden state of tasks */
184     boolean tasksHidden;
185
186     /** The shown/hidden state of events */
187     boolean eventsHidden;
188
189     /** The CalendarDB used to get the currently selected date and to pass on
190     * to subviews */
191     CalendarDB caldb;
192 }

```